DIMP: A Low-Cost Diversity Metric Based on Circuit Path Analysis

Sergi Alcaide[‡], Carles Hernandez[†], Antoni Roca[‡], Jaume Abella[†]

[†]Barcelona Supercomputing Center (BSC), Barcelona (Spain)

[‡]Universitat Politècnica de Catalunya (UPC), Barcelona (Spain)

ABSTRACT

Diversity has been regarded as a desirable property of redundant instances, since it allows circuits to behave differently in front of a given fault. However, while qualitatively diversity is a well-understood concept, usable efficient metrics do not exist to quantify diversity in the context of safety-related systems. In this paper we cover this gap by proposing DIMP, a low-cost diversity metric based on analyzing the paths of the redundant circuits. We relate it to the particular case of automotive microcontrollers implementing lockstep cores and show that it can be successfully used providing relevant information for addressing common cause faults.

1. INTRODUCTION

Safety-related systems must undergo an exhaustive validation and verification (V&V) process before being deployed to prove that their safety goals are met. This relates to collecting enough evidence proving that the risk of physical injury or damage to the health of human beings due to malfunctioning behavior of the system 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.

While these safety goals are defined at the scope of the whole system, they are propagated to its components so that their composition preserves those safety goals. This ultimately translates into attaching to each component a Safety Integrity Level, which determines the type and amount of evidence needed in the V&V process to prove that the component under analysis will meet its specifications. For instance, in the case of the automotive domain, the ISO26262 functional safety standard is used for the V&V process and it defines four Automotive Safety Integrity Levels (ASIL), from A to D, being D the highest integrity level and A the lowest [1]. However, similar standards with analogous classifications of the safety integrity levels can be found in other domains such as avionics (e.g., DO178B/C [21]), railway (e.g., EN50126/8 [3]) and general electronic safety-related systems (e.g., IEC61508 [14]) among others.

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 ASIL C and D functionalities often require some form of hardware redundancy [10, 13]. This is the case, for instance, of the Infineon AURIX microcontroller, where some of its cores execute in lockstep¹ so as to have physical redundancy [13]. However, lockstep execution per se is amenable for faults that can only affect one of the instances at a time. If any fault can produce the same failure on both instances, then redundancy does not suffice. This type of failures (e.g., due to voltage droops) are often referred to as common cause failures and need, not only redundancy, but also *diversity*, so that the manifestations of the fault are different in the different redundant instances and hence, the fault can be detected before becoming a failure.

Diversity can be attained by using different implementations of the same hardware functionality at different abstraction levels. Different benefit/cost ratios are achieved by using diversity at different levels. However, while diversity is well-understood at a qualitative level, it is hard to quantify quantitatively since it relates to the target common cause failure addressed and the abstraction level at which it is considered (i.e. gate level, transistor level). Some efforts have been done to quantify diversity systematically [17–19] but, as we show later, they do not meet the needs of dual modular redundancy to mitigate common cause failures, which have prominent importance in safety-related domains. Therefore, industry faces the complex issue of delivering diverse implementations without practical means to quantify their effectiveness. This issue exacerbates with the increasing complexity of microcontroller designs needed to keep pace with the increasing performance requirements of the safety-related functionalities in different domains such as automotive [4, 6] and avionics [8] among others.

This paper covers this gap by proposing DIMP, a low-cost **D**iversity **M**etric based on circuit **P**ath analysis. In particular, DIMP defines a systematic way to describe and compare circuits at various abstraction levels. Moreover, DIMP can be suited to different common cause failures. In this paper we introduce DIMP and suit it to the particular case of timing failures, as a representative common cause failure², and apply it at the gate-level granularity. Note, however, that

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

DAC '17, June 18–22, 2017, Austin, TX, USA

© 2017 ACM. ISBN 978-1-4503-4927-7/17/06...\$15.00

DOI: <http://dx.doi.org/10.1145/3061639.3062231>

¹Lockstep execution in the case of the AURIX processor consists of having 2 cores executing the same software simultaneously and comparing their outputs to detect any mismatch.

²Timing failures can be caused by many different common cause faults such as voltage droops, crosstalk, and systematic process variations.

this type of failures and abstraction level have been chosen for illustration purposes only. We compare DIMP against existing diversity metrics showing that, in controlled scenarios where specific outputs are expected, DIMP is the only metric delivering the right answer. When applying DIMP to a broader spectrum of circuits, we show that it provides useful results with affordable cost.

The rest of the paper is organized as follows. Section 2 provides some background on safety mechanisms for safety-related circuits and diversity principles. Section 3 presents DIMP. Section 4 presents some evaluation results. Related work is described in Section 5. Section 6 draws the main conclusions of this work.

2. BACKGROUND

This section provides some background on safety mechanisms for safety-related microcontrollers, with particular emphasis in the automotive domain, as well as on some aspects related to diversity.

2.1 Safety Mechanisms

A number of safety mechanisms are included in safety-related microcontrollers to avoid or mitigate fault occurrence and/or fault propagation. For instance, memory management/protection units are often set up to prevent unauthorized accesses to specific memory ranges. Also system and peripheral registers may be protected similarly. Data integrity may require the use of error correcting codes in memory and buses. Execution integrity within cores may require checks to detect invalid opcodes and illegal instructions. Watchdogs are often set up to detect deadline violations so that no task exceeds its time budget. Last but not least, redundancy is often used to detect faults in many components. In this regard lockstep cores are a common solution in the automotive domain [10, 13].

Lockstep execution allows detecting a broad range of faults at the expense of significant power and area costs. Therefore, it is only used for the execution of functionalities in the highest ASIL levels (i.e. C and D). Lockstep can be applied at different granularities that relate to what outputs are compared and when. These granularities are often known as the *sphere of replication (SoR)* [20]. The SoR can be defined at the granularity of pipeline stage, instruction, core or board, for instance [11]. The most common solution due to its moderate cost and flexibility consists of defining the SoR at the core boundaries, like in the AURIX processor [13]. In these designs two cores (leading and trailing cores) together with a checker (that can also be replicated) execute the same functionality simultaneously and all their outputs are systematically compared. Those outputs include: read/written instruction and data memory addresses, data sent to memory and peripherals, interrupts and control signals. On a mismatch the error is reported to a safety management unit, which will take control of the system. Typically, such unit will try to resume the execution from a known-safe state and, if needed and doable, will shift the system to a safe state thus making the safety issue become an availability issue.

Lockstep execution requires synchronized operation in the leading and trailing cores so that they produce the same outputs in the same order and simultaneously (or with a constant lag). This can be achieved by using exactly the same implementation for both cores. However, in the face of some fault types, both cores may produce exactly the same (wrong) output, which will not be detected, thus leading to a common cause failure due to a single-point fault. Therefore,

Figure 1: Different ways of implementing diversity for a given functionality.

some form of diversity in the design and/or implementation of those cores is needed to avoid unreasonable risk due to single-point faults.

2.2 Diversity

Diverse circuit implementations can be obtained in many different forms as sketched in Figure 1. We enumerate some of them for illustrative purposes:

1. One may implement two circuits with completely different architectures. For instance, one could implement an adder using a Carry Lookahead Adder and a Kogge-Stone Adder.
2. Diversity can be obtained using different gate types: we can use a complete set of gates, only NANDs, only NORs, only 2-input NANDs, etc.
3. For a given netlist one may use different transistor geometries.
4. Given specific transistor geometries, one could vary their electrical characteristics such as threshold voltage or other electrical parameters.

Diversity is generally not needed to deal with transient faults or permanent faults due to random process variations inside the circuit since those faults will only affect one of the circuit instances. Therefore, a mismatch will be detected when comparing their outputs. However, other faults may produce exactly the same (wrong) output in both instances if diversity is not used. Among those one can identify voltage droops, transient or permanent faults in their root voltage or clock signals, residual faults due to the limited diagnosis coverage during tests (e.g., even for ASIL D such coverage ranges between 90% and 99.9% for different fault types) and due to the impossibility to attain high coverage for some fault models (e.g., timing faults).

In order to define a suitable diversity metric for a given fault type, one needs first to understand the impact of those faults in circuits to identify what differences across implementations will lead to different outputs in the presence of the same fault. Once this is understood, circuits can be made diverse with respect to the relevant fault types. For instance, different gates consume power differently in front of the same inputs for the circuit. Therefore, voltage droops will have different effect on those gates (e.g., NANDs and NORs) such that, even if the outputs are faulty in both circuit instances, they will be faulty for different levels of voltage and will propagate and combine differently, so that different (wrong) outputs will lead to a fault detection. This will prevent a single-point fault that would, otherwise, lead to a common cause failure.

To the best of our knowledge only one methodology has been proposed to quantify diversity [17–19]. This methodology proposes an abstract diversity metric D that is suited for any fault type. D is defined as:

$$D = \sum_{f_i, f_j} P(f_i, f_j) d_{i,j} \quad (1)$$

where $d_{i,j}$ is the probability of both designs not producing the same erroneous output when a fault is injected in each of them and $P(f_i, f_j)$ is the probability of having faults f_i and f_j at each of the circuit instances. Unfortunately, deriving fault probabilities for any potential fault pair affecting the two circuit instances of the redundant system is rather unpractical in the general case. In the particular case that faults are equally probable, D can be computed systematically using the expression given in [17–19]:

$$D_{INJ} = \frac{1}{m} \sum_{i,j} d_{i,j} \quad (2)$$

where m stands for the number of fault pairs injected in both circuit instances and $d_{i,j}$ is again the probability of both designs not producing the same erroneous output.

We refer to that metric such as D_{INJ} since it builds upon injecting faults in the two circuit instances. D_{INJ} operates in the following way: in front of two circuit instances that are not absolutely identical, the same number of faults is injected randomly in both instances.

D_{INJ} reflects how reliable are the two instances in front of random faults. However, common cause failures are produced due to the same fault. Therefore, assuming random independent faults in both circuit instances does not match the fault model of interest for common cause failures. In fact, if we consider two almost-identical implementations of the same circuit with 10% error probability when a fault occurs, D_{INJ} will obtain faults in both circuits simultaneously around 1% of the times, and since faults have been injected in random locations, they will likely lead to different wrong outputs, so that $D_{INJ} \approx 0$. Instead, what we expect is that the same fault (e.g., a voltage droop) makes both instances to fail 10% of the times, and in those cases very likely they will produce the same wrong output. Thus, one could conclude that those circuits are not diverse at all w.r.t. that fault model. Note that the fact that 90% of the faults do not produce a fault should not be taken into account in terms of diversity since it relates to the robustness of the circuits rather than to their diversity.

Therefore, an appropriate diversity metric is needed for safety-relevant circuits.

3. DIMP: A DIVERSITY METRIC BASED ON CIRCUIT PATH ANALYSIS

In this section we present DIMP, a low-cost diversity metric based on circuit path analysis. First, we introduce the rationale behind DIMP. Then we describe its specific implementation for some common cause failures and gate-level circuit descriptions.

3.1 Rationale behind DIMP

DIMP builds upon the idea that lack of diversity occurs when a given input signal I_i propagates to a given output signal O_j through *similar* devices in both circuit instances. Therefore, diversity needs to be quantified taking into account whether the same devices are traversed in the same order from I_i to O_j in both circuit instances. For that purpose, it is required to identify what parts of the circuit are susceptible to a given fault type and compare them across circuit instances taking into account separately all pairs $\langle I_i, O_j \rangle$.

```

1  $DIMP = 0$ 
2  $MaxDIMP = 0$ 
3 For  $i = 1$  to  $N$ 
4   For  $j = 1$  to  $M$ 
5      $Paths_{i,j}^1 = \bigcup \left( Paths \left( I_i^1, O_j^1 \right) \right)$ 
6      $Paths_{i,j}^2 = \bigcup \left( Paths \left( I_i^2, O_j^2 \right) \right)$ 
7     While  $Paths_{i,j}^1 \neq \emptyset$  and  $Paths_{i,j}^2 \neq \emptyset$  do
8       Take  $p_k^1 \in Paths_{i,j}^1$  and  $p_l^2 \in Paths_{i,j}^2$ 
          with highest  $overlap(p_k^1, p_l^2)$ 
9        $DIMP = DIMP + weight(p_k^1, p_l^2) \cdot (1 - overlap(p_k^1, p_l^2))$ 
10       $MaxDIMP = MaxDIMP + weight(p_k^1, p_l^2)$ 
11      Remove  $p_k^1$  from  $Paths_{i,j}^1$ 
12      Remove  $p_l^2$  from  $Paths_{i,j}^2$ 
13    Endwhile
14     $RemainingPaths_{i,j} = Paths_{i,j}^1 \cup Paths_{i,j}^2$ 
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  $\frac{DIMP}{MaxDIMP}$ 

```

Figure 2: Pseudocode of a realization of DIMP.

Once all irrelevant parts of the circuit have been filtered out and those devices traversed to get from I_i to O_j kept, then a comparison across both circuit instances needs to be performed. Such comparison needs to weight each path from I_i to O_j and each device in each path conveniently to reflect the weight of each path and each device in the manifestation of a given fault at the output O_j . For instance, fault types manifesting as timing faults in the circuits may be more vulnerable to longer paths than to shorter paths. In that case, it is important reflecting that short paths are less likely to lead to erroneous outputs and thus, their diversity is less relevant than that of longer paths.

3.2 A Realization of DIMP

Next we describe a particular realization of DIMP intended for fault types including voltage droops, residual faults escaping diagnosis coverage and permanent faults in the root of the power network of the circuit instances. The pseudocode of DIMP for this realization is shown in Figure 2. The algorithm works as follows. First, we iterate over all pairs $\langle I_i, O_j \rangle$, as shown in lines (3-4), where N stands for the number of input signals and M for the number of output signals. For each pair $\langle I_i, O_j \rangle$ we regard as relevant all paths from I_i to O_j and all devices in each path. Since we work at gate-level in this particular realization, the devices correspond to all gates traversed in a given path. Then, $Paths_{i,j}^1$ stands for the list of all paths from I_i to O_j in the first instance of the circuit, whereas $Paths_{i,j}^2$ includes the same information but for the second instance of the circuit, as shown in lines (5-6).

Then, we iterate over both sets of paths as long as none of them is empty, see line (7). Out of the remaining paths in each set, we pick one from each set so that their overlap is maximized, see line (8). Here we define $overlap(p_k^1, p_l^2)$ as the number of gates that repeat across paths in the same order, even if some other gates are interleaved, divided by the total number of gates in both paths. For instance, if $p_k^1 = \{NAND2, NOR2, NOT, XOR2\}$ and $p_l^2 = \{NAND2, NOR2, XOR2\}$, then $\{NAND2, NOR2, XOR2\}$ repeat, so $overlap(p_k^1, p_l^2) = 6/7$ since 6 gates repeat in both paths out of the 7 they have. Note that other definitions of $overlap(p_k^1, p_l^2)$ are possible. We increase $DIMP$ with the non-overlap (so the diversity) weighted by the relevance of the corresponding

paths, see line (9). We define $weight(p_k^1, p_i^2)$ as the relative weight of those paths w.r.t. the other paths of the circuit. Since we target timing faults in this particular implementation, we define $weight(p_k^1, p_i^2)$ as the maximum gate count of those paths. In the case of residual faults, one could simply make $weight(p_k^1, p_i^2) = 1$ for all paths. However, devising appropriate weights for different fault types is beyond the scope of this paper.

In the algorithm, *MaxDIMP* stands for the maximum potential diversity, so it is updated accordingly in line (10). Then we simply remove those paths considered from their corresponding sets in lines (11-12).

Whenever at least one of the sets of paths is empty, we process the remaining ones analogously as shown in lines (14-20). In this case $weight(p_q, \emptyset)$ corresponds to the weight of path p_q and $overlap(p_q, \emptyset)$ is 0.

Finally, we return the diversity measure as the actual accumulated diversity (*DIMP*) divided by the maximum potential diversity of the circuit (*MaxDIMP*).

Note that two identical circuit instances will obviously lead to diversity 0 since we will match identical paths all the time. Analogously, two completely distinct circuit instances (e.g., one implemented with NOR gates and another with NAND gates) will return diversity 1 since their overlap will be always 0. Intermediate values will be later discussed in the context of specific examples in the evaluation section.

4. EVALUATION

In this section we present the evaluation framework, the results of applying the proposed diversity metric, and the experimental validation of the results obtained.

4.1 Experimental Framework

For computing *DIMP* we have implemented in C++ the algorithm presented in 2. Our algorithm takes as inputs all the paths of a given circuit implementation. Circuit paths have been obtained after circuit synthesis from timing reports listing all paths. We have used some circuits from the ISCAS bench suite [2] to assess the diversity metric and for illustration purposes we have used an 8-bit full-adder. Other (larger) ISCAS benchmarks have not been considered because of the computational cost of *D_{INJ}*, needed for comparison purposes, although our methodology (*DIMP*) was applied successfully. In order to have different gate implementations of each circuit we have used synthesis optimization for different delay targets, being *0.3ns* the most demanding *1.0ns* the most relaxed constraint.

We have also computed the diversity of the test circuits using the systematic implementation of the diversity metric given in [18]. For that purpose we have followed *D_{INJ}* equation and injected stuck-at faults following the methodology given in [9]. Faults have been randomly injected at the gate-level circuit implementation exploring all possible input combinations. For each input we have injected 10,000 different stuck-at faults. To determine whether the fault produces an error, we compare the outputs against the golden run.

Finally, for verification purposes we have computed the diversity with respect to voltage droops with SPICE. To do so, we have used an 8-bit full adder as circuit under test since it allows us to easily build completely different implementations that are functionally equivalent. For every circuit implementation we inspect the outputs and analyze whether the faults caused by voltage droops produce the same error manifestation or not.

Table 1: Diversity results computed injecting random faults.

	0.3 ~ 0.6	0.3 ~ 1.0	0.6 ~ 1.0
s1196	0.999999	0.999998	0.999986
s1238	0.999999	0.999999	0.999999
s1488	0.998937	0.999332	0.998904
s1494	0.999998	1	0.999962
s27	1	1	1
s298	0.996409	0.999571	0.996084
s382	1	0.998712	1
s386	0.999878	1	0.99947
s400	0.99855	0.998784	0.9988
s444	1	0.999234	1
s953	1	1	1

4.2 ISCAS

To test the proposed diversity metric we have implemented ISCAS circuits using three different time constraints: *0.3ns*, *0.6ns* and *1.0ns* in order to produce 3 diverse implementations of the same circuit. Table 1 shows results of diversity when using *D_{INJ}*. Every column shows the diversity computation between two of the three implementations available. As shown in the table, with this metric diversity is roughly 1 for all circuits and implementation pairs. In principle, this would mean that the circuit implementations are very diverse no matter the target delay used for their synthesis. However, if we analyze these circuits it is easy to observe they have significant similarities, specially between those paths that are not time-critical, where synthesis optimization introduces fewer modifications. Therefore, one would expect achieving values of diversity not so close to 1. This occurs because *D_{INJ}* is suitable for random faults and, the probability that two random faults affecting to each of the circuit instances produce the same error manifestation, is almost zero. In fact, this metric measures how robust a given circuit is w.r.t. random faults rather than how diverse two given circuit implementations are. Thus, if we apply this metric to two circuit instances that are almost identical random faults are extremely unlikely to produce the same erroneous output in both circuit instances, so we will obtain that $D_{INJ} \approx 1$. Instead, the actual diversity of those two instances should be close to 0.

Table 2 shows the result of applying *DIMP* to the different circuit implementations. With our metric diversity ranges from 0.6 to 1 better representing differences across different circuit implementations. We observe a similar behavior across the different implementations of a given circuit. With *DIMP* the greater diversity values are achieved between the implementations targeting *0.3ns* and *1.0ns*. This is an expected behavior since we compare implementations for the most relaxed (*1.0ns*) and most demanding (*0.3ns*) optimization targets. Additionally, it is noteworthy to mention that *DIMP* has been suited for timing failures and we compare the two implementations with largest difference in the propagation delay. However, there are two circuits for which this trend is not followed: **s382** and **s27**. We have inspected the gate implementations of these two circuits and have realized that the utilization of some gates in some paths produces some cascade effects on the automatic choice of other gates in the same path. Those other gates, therefore, may repeat across implementations targeting *0.3ns* and *1.0ns*, but not for the implementations targeting *0.6ns*.

Finally, we have also observed that the circuit pair providing the lowest diversity values is the one given by *0.6ns* and *0.3ns*. For most circuits the delay without optimizations is close to *0.7ns*. This results in the synthesis tool requiring

Table 2: DIMP’s results.

	0.3 ~ 0.6	0.3 ~ 1.0	0.6 ~ 1.0
s1196	0.87885	0.990901	0.989536
s1238	0.881473	0.980223	0.977516
s1488	0.898724	0.951677	0.815758
s1494	0.884235	0.98916	0.973529
s27	1	0.949367	1
s298	0.77582	0.991486	0.981879
s382	0.678801	0.873572	0.901327
s386	0.933984	0.960784	0.791227
s400	0.60297	0.880554	0.880322
s444	0.601725	0.879051	0.827288
s953	0.625419	0.892768	0.885153

Table 3: DIMP’s results.

	NAND ~ AND	NAND ~ FA2	NAND ~ MIX
FA	0.917299	1	0.41

significant timing optimization effort for 0.3ns and 0.6ns, which end up using the same gates relatively often. For the 1.0ns target, instead, optimizations focus on reducing power consumption, thus leading to more diverse implementations w.r.t. the other targets.

4.3 SPICE validation

In the previous section we have reasoned about the proposed metric *DIMP* and found it produces results in line with the expectations based on visual inspection. In this section, in order to increase the confidence on the proposed *DIMP* metric, we use SPICE and four different implementation of an 8-bit full adder (FA). The four FA implementations are: a regular full adder implemented with AND gates (FA-AND), the same circuit but implemented with NAND gates (FA-NAND), a FA implementation using a completely different architecture (FA2), and a full adder that is build using a 4-bit NAND FA and a 4-bit AND FA (FA-MIX). We have simulated these circuits with SPICE covering all the space of possible operand inputs and considered several voltage points below the nominal one (1V). With SPICE simulations, we compute the percentage of outputs providing different results for different circuit pair combinations. Note that results with SPICE are not analogous to the ones our metric provides since, while 1 represents the maximum diversity, 0 represents that either the circuits are identical or that the outputs provided by the circuits are correct. However, these results help us understanding if *DIMP* is able to capture the diversity of error manifestations caused by voltage droops.

Figure 3 shows the percentage of outputs providing different results for different circuit pair combinations. For the lowest simulated voltage (0.6V), where a large fraction of errors is expected the NAND-AND and NAND-FA2 configurations provide very diverse erroneous outputs, as we increase voltage, failures decrease. Thus, the number of diverse outputs naturally decreases due to both implementations providing the same (correct) output. This is in line with the *DIMP* results shown in Table 3. As shown in the table these two circuit pairs present high diversity since they are based on significantly different implementations. SPICE simulations confirm that this topological diversity also translates in diverse error manifestations. With *DIMP*, NAND and AND-based implementations of the FA show diversity 1 w.r.t. the alternative full-adder implementation FA2, and a significant diversity (≈ 0.92) between FA-NAND and FA-AND, whose circuit implementations repeat some NOT gates (i.e. NAND-1 is a NOT, and AND implementa-

Figure 3: Diversity for different full adder circuits implementations.

Figure 4: Computational complexity. Square brackets show the range of the depth of the paths.

tion needs some NOT gates). This behavior is also reflected with the SPICE simulations where highest diversity always involves FA2 and either FA-NAND or FA-AND. Finally, the pair NAND/MIX is the one providing lower diversity since roughly half of the circuit is identical in both replicas. Again, SPICE results are in line with *DIMP* and show the lowest diverse error manifestations for the NAND/MIX setup.

4.4 Computation complexity

An important property of our metric is that it can be computed systematically and that its computational costs are low. Figure 4 shows the amount of time we require for computing *DIMP* for the different synthetic circuit implementations. As shown in the plot computational costs are in general low, and only for some circuit configurations consisting in a significant number of paths with few inputs and outputs computations take up to few hours. We have also confirmed measuring the computation time of ISCAS benchmark since only 2 circuits required times above one hour. Instead, the computation of D_{INJ} may require several days of computation for some of the circuits. Moreover, the absence of a systematic procedure to relate D_{INJ} results with diversity for any fault model for a given circuit implementation, challenges its application for the problem addressed in this paper.

While a computational cost of few hours is not negligible, note that diversity needs to be quantified once for each pair of diverse implementations during product development and can be done hierarchically so that different parts (e.g., each functional unit block, FUB) of a microcontroller can be evaluated separately. Thus, FUBs along with their diversity quantification can be reused, and multiple FUBs can be easily evaluated in parallel.

5. RELATED WORK

Error detection and correction codes have been used widely in memory devices and caches of safety-relevant systems [5]. Analogously, some form of redundancy has been considered

for the remaining processor components [10, 12, 15]. Solutions based on lockstep execution have been shown to fit well safety-relevant microcontrollers and have been deployed in a number of microcontrollers. This is particularly true in the automotive domain [7, 10, 13], where processors deliver 2 or 3 user-visible cores together with some other cores used for lockstep execution only. However, these solutions are effective to deal with those faults that do not lead to common cause failures. Instead, additional mitigation measures are needed to prevent that a single-point or a residual fault leads to a common cause failure.

Diverse designs have been considered in academia without relating them to common cause failures [16, 22]. Industry, however, have employed diverse designs to effectively mitigate common cause failures, although this fact has been scarcely disseminated [13, 23]. Unfortunately, industry faces the difficult challenge of quantifying whether two circuit instances are diverse enough to avoid unreasonable risk. To the best of our knowledge only Mitra et al. [17–19] have proposed a systematic and quantitative approach to measure diversity. Unfortunately, such diversity metric is suited for random faults affecting circuit instances independently, so it cannot be used for those faults leading to common cause failures. This paper covers this gap and introduces DIMP, a suitable diversity metric for common cause failures in safety relevant microcontrollers.

6. CONCLUSIONS AND FUTURE WORK

Lockstep execution is often used in safety-related microcontrollers to meet the safety requirements for the highest integrity levels. However, while lockstep has been proven efficient to detect independent faults across redundant cores, it is unable to detect those faults leading to common cause failures unless diverse designs are used. This poses a new challenge for industry: how to quantify diversity to assess that unreasonable risk is avoided.

This paper tackles this challenge by providing DIMP, a low-cost diversity metric based on circuit path analysis that systematically quantifies the degree of diversity of different circuit implementations. DIMP is a flexible metric that can be suited to different abstraction levels (i.e. gate level, transistor level) and different fault types. We provide a specific realization of DIMP for timing faults and gate level abstraction. Our evaluation shows the effectiveness and low cost of DIMP to quantify diversity, thus proving that it is a powerful tool for chip manufacturers in safety domains.

While DIMP is the first approach towards an automated quantification of diversity against single-point failures, as needed for verification in automotive safety-related systems, a number of steps need still to be carried out for the industrialization of DIMP: (i) relating DIMP results with industrial-relevant examples where reference diversity values can be obtained a priori and hence, DIMP results can be assessed and the method tuned for the specific fault model addressed. This is part of our future work. (ii) Integrating diversity quantification as part of ISO26262 procedures in the next releases of the standard to provide automotive microcontroller industry with references for system assessment against the standard [13, 23]. This is a joint effort involving a number of (mostly) automotive industrial partners that have explicitly requested these modifications in the standard.

Acknowledgments

This work has been partially supported by the Spanish Ministry of Economy and Competitiveness (MINECO) under

grant TIN2015-65316-P and the HiPEAC Network of Excellence. Jaume Abella has been partially supported by the MINECO under Ramon y Cajal postdoctoral fellowship number RYC-2013-14717. Carles Hernández is jointly funded by the MINECO and FEDER funds through grant TIN2014-60404-JIN.

7. REFERENCES

- [1] . *ISO/DIS 26262. Road Vehicles – Functional Safety*, 2009.
- [2] F. Brglez, D. Bryan, and K. Kozminski. Combinational profiles of sequential benchmark circuits. In *ISCAS*, May 1989.
- [3] CENELEC. *EN50128. Railway Applications: Software for Railway Control and Protection Systems*, 2011.
- [4] R.N. Charette. This car runs on code. In *IEEE Spectrum online*, 2009.
- [5] C.L. Chen and M.Y. Hsiao. Error-correcting codes for semiconductor memory applications: A state of the art review. *IBM Journal of R&D*, 28(2):124–134, 1984.
- [6] P. Clarke. Automotive chip content growing fast, says gartner, 2010. <http://www.eetimes.com/electronics-news/4207377/Automotive-chip-content-growing-fast>.
- [7] C. Cullmann et al. Predictability considerations in the design of multi-core embedded systems. In *ERTSS*, 2010.
- [8] G. Edelin. Embedded systems at thales: the artemis challenges for an industrial group. In *Presentation at the ARTIST Summer School in Europe 2009*, 2009.
- [9] J. Espinosa et. al. Analysis and rtl correlation of instruction set simulators for automotive microcontroller robustness verification. In *DAC*, 2015.
- [10] Freescale Semiconductor. Qorivva MPC5643L microcontroller data sheet. rev. 9, 2013.
- [11] C. Hernandez and J. Abella. LiVe: Timely error detection in light-lockstep safety critical systems. In *DAC*, 2014.
- [12] IBM. *PowerPC 750GX Lockstep Facility. Application note*, 2008.
- [13] Infineon. AURIX Multicore 32-bit Microcontroller Family to Meet Safety and Powertrain Requirements of Upcoming Vehicle Generations. <http://www.infineon.com/cms/en/about-infineon/press/press-releases/2012/INFATV201205-040.html>.
- [14] International Electrotechnical Commission. *IEC61508. Functional safety of electrical/electronic/programmable electronic safety-related systems*, 2010.
- [15] R.E. Lyons and W. Vanderkulk. The use of triple modular redundancy to improve computer reliability. *IBM Journal of R&D*, 6(2):200–209, 1962.
- [16] S. Mitra and E.J. McCluskey. Design diversity for concurrent error detection in sequential logic circuits. In *VTS*, 2001.
- [17] S. Mitra, N.R. Saxena, and E.J. McCluskey. Techniques for estimation of design diversity for combinational logic circuits. In *DSN*, 2001.
- [18] S. Mitra, N.R. Saxena, and E.J. McCluskey. A design diversity metric and analysis of redundant systems. *IEEE Transactions on Computers*, 51(5), 2002.
- [19] Subhasish Mitra, Nirmal R. Saxena, and Edward J. McCluskey. Efficient design diversity estimation for combinational circuits. *IEEE Trans. Comput.*, 53(11):1483–1492, November 2004.
- [20] S.K. Reinhardt and S.S. Mukherjee. Transient fault detection via simultaneous multithreading. In *ISCA*, 2000.
- [21] RTCA and EUROCAE. *DO-178B / ED-12B, Software Considerations in Airborne Systems and Equipment Certification*, 1992.
- [22] F. Salewski and S. Kowalewski. Achieving highly reliable embedded software: An empirical evaluation of different approaches. In *SAFECOMP*, 2007.
- [23] Texas Instruments. Hercules microcontrollers: Real-time MCUs for safety-critical products, 2011. White Paper.