Whose fault is it? Advanced techniques for optimizing ISO 26262 fault analysis

Avidan Efody
Mentor Graphics, Corp.
10 Aba Eban Blvd.
Herzilya 46120, Israel
avidan_efody@mentor.com

Abstract-Shrinking nodes and reduced supply voltages make transient faults due to electromagnetic interferences a growing concern for mission critical ASICs and FPGAs. To address this risk, the ISO 26262 safety automotive standard requires that the impacts of transient faults on safety goals are rigorously analyzed[1]. Such analysis is far from trivial, first and foremost due to the practically infinite number of fault and state combinations that could happen in a component's life cycle. This paper deals with transient fault analysis towards ISO 26262 certification. First we suggest a way to estimate ISO 26262 required metrics with a user specified level of accuracy using statistical sampling of transient faults. We then propose a technique that reuses existing regression results in order to minimize the resources required to analyze faults in both combinatorial and sequential elements.

I. CONTEXT

Shrinking nodes and reduced supply voltages make transient faults due to electromagnetic interferences a growing concern for mission critical ASICs (Application Specific Integrated Circuit) and FPGAs (Field Programmable Gate Array). Unexpected changes to data, configuration, state or control elements within the design can lead to catastrophic failures if not taken into account and addressed upfront, either by additional safety mechanisms or through analysis that shows their risk to be sufficiently low within the component's life-cycle.

The ISO 26262 automotive safety standard requires that the probabilities of random hardware failures are rigorously analyzed and quantified via a set of objective metrics to allow for external auditing. These metrics include the so called **architectural metrics** which assess the adequacy of the safety mechanisms chosen[2], and their ability to prevent faults from reaching safety critical areas by detecting or correcting them. Although transient faults are only one reason that could lead to random hardware failures, quantifying their impact as part of the architectural metrics evaluation is one of the most challenging aspects of the certification process, due to the immense number of state and fault combinations.

In cases where any of the architectural metrics fail to meet the criteria defined for the product's Automotive Safety Intergrity Level (ASIL), re-evaluation of the component safety concept possibly followed by improvement of existing safety mechanisms or introduction of new safety mechanisms is mandated by the standard. Since transient fault related metrics are usually obtained from gate level netlists, sometimes even after place and route, they are usually not available until late in the design cycle. At this stage, conceptual re-evaluation and substantial hardware modifications together with the verification required might have severe impact on schedule.

An analogy to the implementation of a power concept can highlight important similarities and one important difference. Error rates per block can be allocated, just like power budgets, during the architectural exploration phase. Blocks will then be translated into RTL (Register Transfer Level) and GL (Gate Level), at which point their actual error rate or power consumption, can be accurately measured, and compared against the high level target values. If a problem is discovered at this stage, the solution space is much more limited, and the saving per solution is usually much smaller. The main difference between the two flows is that while it may be ok for a block's power consumption to be very close to the initial target, with error rate this will mean a significant increase in analysis efforts. This stems

from the fact that to comply with ISO error rates must be "proven". A proof for external purposes, is a lot harder to achieve.

The closer the metrics are to the criteria defined, the higher their accuracy is required to be. The higher the accuracy required, the more effort must be invested in fault analysis. Hence there is a trade-off between design cost and analysis cost. If a design's safety concept is planned upfront with a significant margin from the criteria defined, then analysis could be relaxed. This would mean additional design cost, at the expense of analysis cost. The opposite applies in the same way – if a design is planned to meet targets exactly, analysis will need to become much more sensitive, which will result in an extra analysis cost. As we will see in the sections below, this extra cost in analysis will manifest itself in a growth in the number of faults that need to be analyzed.

In this paper we tackle the transient fault coverage space challenge from two sides. First we explain how this practically infinite space can be managed, and a limited number of faults can be used to obtain the relevant metrics at the required accuracy. We then describe a technique allowing the limited sample chosen to be run at reduced cost in time and resources.

II. Fault sampling

The statistical method is based on analyzing a sample of faults chosen at random from the population in order to estimate the various rates required by the standard. It consists of the following 6 main steps. This section will go into detail about all of these steps, except for the actual fault analysis which will be described in the next section. An example calculation will be provided at the end of the section as well.

1. Choose fault sample size N
2. Pick N random (signal ,cycle) pairs from regression
3. Analyze results of a fault for each of the N (signal,cycle) pairs
4. Approximate ISO 26262 required rates +/- error
5. Approximate ISO 26262 required metrics +/- error.
6. Close loop:
    a. If clearly meeting criteria -> done
    b. If clearly not meeting criteria -> reevaluate safety concept
    c. If error too large to determine -> increase sample

A. Choose fault sample size N

At the time when we start the fault analysis process, the only information available to us is the **FIT** (Failures In Time) target values set for the component during the planning phase (1 FIT target value means the component should not fail more than once in billion hours). These FIT target values could be divided by the base FIT of the component to determine the expected **de-rating**. The base FIT takes the worst case assumption that every fault occurring in any gate in the design, will result in a safety critical failure. However, some faults will usually be masked before they mature into failures, and their ratio from the entire fault population is called the de-rating factor. If we divide our target FIT by the base FIT we can calculate the expected de-rating i.e. how many faults we expect to be masked.

This expected de-rating could be used to provide an initial guess about sample size: to provide a reliable measurement the sample size should be inversely proportional to the de-rating rate, to allow some faults to mature into failures. Once a first iteration of steps 1-6 is performed, if results considerably deviate from expectations, or if the accuracy of the results doesn't allow sufficient confidence in matching against the criteria, the sample size should be increased.

B. Pick N random (signal ,cycle) pairs from regression

In a typical design, different quantities of different kinds of cells are being used, and these cells differ in their FIT rate. In the typical case, N will be large enough to get a representative percentage of each different cell. However, it

is recommended to map the cells used and their corresponding FIT rates, and make sure extreme outliers get a fair share. For example, if a cell that is being used only rarely, has a FIT rate that is orders of magnitude higher than other cells, it might have an impact on the component FIT rate and hence the sample should include it.

One set of elements that might be used only a few times compared to other cells but still have an impact on overall FIT are memories, especially when not protected by ECC. In cases when there are such unprotected memories their FIT should be calculated and if found significant, the representation of their output signals in the samples should be artificially increased. Memory outputs should basically be seen as accumulating the FIT rates of all memory cells, which though not visible in the gate level netlist, will still exist in the real design.

The run time has no impact on the FIT rate of cells, and hence all cycles are equivalent, and the sample will be representative for any N.

Any faults that can't impact safety critical areas or can do so only in conjunction with more than one fault are defined by the standard as **safe faults**. The signals pool from which the signals are sampled could include signals that are not connected or have a read only connection to safety critical areas. If such signals are included in the pool, the rate of safe faults could also be determined as part of the overall process for any of the other rates. It is also possible to leave these signals out and calculate the safe fault rate based on their percentage in the number of all signals in the design. In this paper we will assume that the 2nd method is used, and will not refer to faults that happen on signals not connected.

C. Analyze results of a fault for each of the N (signal,cycle) pairs

At this stage the impact of a bit flip in the given signal at the given cycle are analyzed. As mentioned above, the second part of this paper is dedicated to analyzing the impacts of faults in combinatorial logic (i.e. outputs of gates that are not part of FF), and sequential logic.

D. Approximate ISO 26262 required rates +/- error

Analysis should classify each of the N faults as **safe fault**, **Single Point Fault (SPF), Residual Fault (RF), Multi Point Fault (MPF), detected fault** or **latent fault.** The definitions for each of these types of faults can be found in section 1 of the ISO standard[3]. For most other fault models (for example stuck-at faults), the standard requires that analysis distinguishes between detected fault and latent fault, where a detected fault is basically one that is flagged by a safety mechanism. For transient faults, however, this distinction doesn't apply since most of them are very short in duration, and could disappear from the system well before propagating to any safety mechanism[4]. Hence the categories that apply for transient faults are safe, single point, residual, and multi point.

The number of faults for a given type, divided by N, is the approximate rate. This rate is an approximation because the analysis was performed on a sample of an infinite population, and other samples could give different results. The probability of the actual rate to be within a given range of that value is dependent on the standard error, which in its turn is dependent on the root mean square of N.

For example, if the approximation of the SPF rate λ_{spf} is given by $\hat{\lambda}_{spf}$ then its standard error is given by [5]:

$$\sigma_{spf} = \sqrt{\frac{\hat{\lambda}_{spf}(1 - \hat{\lambda}_{spf})}{N}}$$

The probability of the real value being λ_{spf} is given by [6]:

$$P\left(\lambda_{spf}, \hat{\lambda}_{spf}, \sigma_{spf}\right) = \int_{x}^{\infty} \frac{1}{\sqrt{2\pi}\sigma_{spf}} e^{-\frac{1}{2}\left(\frac{\lambda_{spf}-\hat{\lambda}_{spf}}{\sigma_{spf}}\right)^2} dt$$

This means, for example, that the probability of λ_{spf} being bigger than the approximated value ($\hat{\lambda}_{spf} + 5\sigma_{spf}$) is 0.9999997 and the probability of it being bigger then ($\hat{\lambda}_{spf} + 7\sigma_{spf}$) is 0.999999999998 [7]. Since N can be increased to make σ_{spf} smaller, this means that users can fine tune the approximation of the various rates to the accuracy and the level of confidence they require.

E. Approximate ISO 26262 required metrics +/- error

Once the values are approximated they can be used to calculate the required ISO 26262 metrics.

SPFM

The Single Point Fault Metric (SPFM) is defined as[8]:

$$SPFM = 1 - \frac{\lambda_{spf} + \lambda_{rf}}{\lambda}$$

Where:

- λ - is the FIT for the entire component (calculated by multiplying all gates by their FIT)
- λ_{spf} – is the above calculated single point fault rate
- λ_{rf} – is the above calculated residual fault rate

The standard error for this expression will be equal to the sum of standard errors for λ_{spf} and λ_{rf} divided by λ.

LFM

As mentioned above the standard doesn't require that transient multiple point faults are further divided into latent/detected faults. Hence the Latent Fault Metric, required for other fault types, isn't required for transient faults.

PMHF

The Probabilistic Metric for Random Faults can be calculated by summing single point fault rate and residual fault rate[9]:

$$PMHF = \lambda_{spf} + \lambda_{rf}$$

Where:

- λ_{spf} – is the above calculated single point fault rate
- λ_{rf} – is the above calculated residual fault rate

Diagnostic Coverage

The diagnostic coverage with respect to residual faults is defined as[10]:

$$DC_{with\ respect\ to\ residual\ faults} = \left(1 - \frac{\lambda_{rf}}{\lambda}\right) * 100$$

Where:

λ - is the FIT for the entire component

λ_{rf} – is the above calculated residual fault rate

F. Close loop

At this stage approximate values for the various metrics need to be checked against the target values. This can lead to one of three options. The first is that the metrics clearly meet the criteria, in which case the process is done. The second is that the level of accuracy of the results doesn't allow to clearly determine if the target values are met or not – this will usually be the case when the calculated metrics lay close to the target values. In this case the number of samples will have to be increased. The last option is that metrics clearly miss the criteria in which case the safety concept needs to be re-evaluated and safety mechanisms improved or added.

III. Fault analysis

Fault analysis is aimed at determining the percentage of faults that turn into failures. This percentage is called a de-rating factor. Fault de-rating could be done at different abstraction levels. The FIT rate per-gate is usually already de-rated by a factor reflecting process maturity, and could be further de-rated if a limited range of temperatures or higher supply voltage is being used. After place and route, faults happening at FF inputs could be de-rated by considering their timing, as a fault is more likely to turn into error if it happens at specific points in the sampling window[11]. In the following sections we will focus on combinatorial and sequential logic de-rating, which reflect the fact that a fault, even when it turns into an error, could be masked by downstream logic and FFs before reaching any safety critical area. At an even higher abstraction level one could think of use-case specific de-rating which would consider only the parts of a design that are active for a given use-case and de-rate all other parts. As can be expected, the higher the abstraction level at which the de-rating takes place, the more significant it usually is, since it takes into account real knowledge about the design's operation.

ISO 26262 recommends that transient fault analysis is carried out on a gate level netlist[12], and on RTL only if 1-1 mapping from RTL to gate level netlist could be established for the analyzed faults. As mentioned above, for some kinds of de-rating which apply to transients timing information could be useful. However, for logic and sequential de-rating it is not, which means that the gate-level netlist could be run without SDF (Standard Delay Format).

The traditional method for fault analysis is "fault injection". At the level of gate-level without timing fault injection usually consists of fully re-running one of the regression tests, flipping a bit at some point in time, then analyzing the results to see if any of the signals defined as safety critical (outputs, state-machines) had changed in any way. The analysis is usually carried out by waveform comparison against a reference model with no fault injected. To determine the de-rating factor with reasonable accuracy a large enough number of faults must be injected which could be resource and time consuming. In this section we propose a technique for fault analysis that makes much better use of existing regression results.

G. Technique overview

The first step in the proposed technique is to distinguish between faults occurring in combinatorial logic and faults occurring in memory elements. If the signal in the (signal, cycle) pair chosen is the output of a logical gate than we consider the fault to occur in combinatorial logic. If the signal in the (signal, cycle) pair chosen is the output of memory or FF we consider the fault as occurring in memory element.

Faults that occur in combinatorial logic have a high chance of being masked within the cycle in which they happen, while faults that occur in memory elements might persist for an indefinite number of cycles until the memory element is being rewritten. For faults that happen in combinatorial logic we propose to first execute an analysis of the current cycle, at the end of which it is known if the fault has reached a memory element or has been masked. A combinatorial logic fault that has reached memory element(s) can be considered from that moment on equivalent to memory element fault, and is analyzed in the same way. The only difference is that a combinatorial logic fault might impact more than one memory element, whereas memory element fault will impact in the initial cycle only one memory element. Figure 1 shows the scope of the two different kinds of analysis.

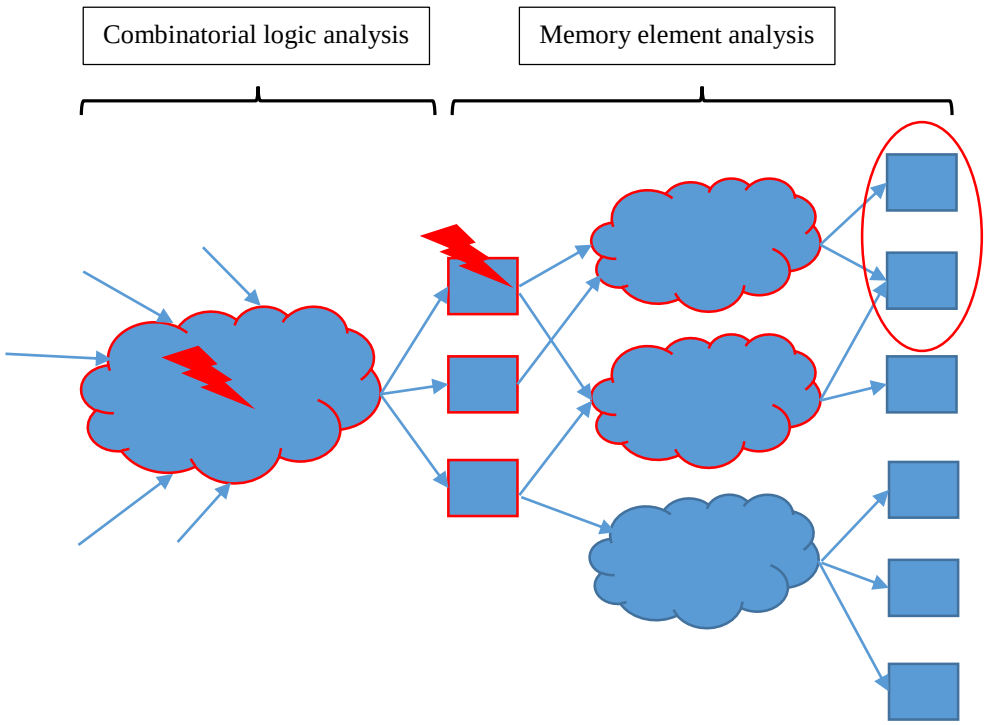

Figure 1: combinatorial logic fault analysis and memory element fault analysis scope

H. Combinatorial logic fault analysis

In almost any combinatorial logic beyond single “not” or “buffer” gates, some faults will be masked by combinatorial logic before being stored in a FF or memory at the end of the cycle. For a given fault injected at a given cycle on a given signal, and provided that all other signals feeding into this logic have been logged, it is easy to recalculate the result at the end of the cycle and checking if any of the inputs to the memory elements that sample this logic has changed. Figure 2 shows an example of a value that will change some inputs to memory elements and a value that will be masked.

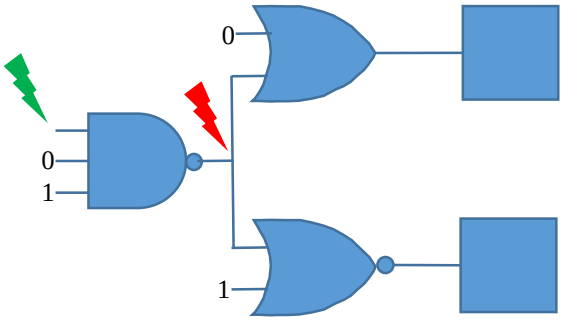

Figure 2: A masked transient fault (in green) and a propagating transient fault (in red)

If analysis is run on a platform that supports X values on signals (i.e. simulation rather than emulation/prototyping), injecting X at the injection point, will save the need to compare the values stored by memory elements to the original values. If the X propagates to the memory element inputs the fault has propagated. When using this technique, however, care has to be taken to make sure that the cells propagate X's when they should, and X-optimism is not converting some of the X's into known values. Although this is less of a problem when working at GL than at RTL, sometimes this might require a replacement of the models inside the cells with fault-analysis friendly models. Note that if the user has a way of reproducing emulation or prototyping results from a given point in time on simulation, then this technique can be used on emulation or prototyping results as well.

As mentioned above, for faults that propagate to a memory element the fault analysis needs to be continued in the same way as described in the next question for memory elements. Faults that did not propagate to a memory element could either be "multi-point faults" or "safe faults". Since distinguishing between these two types doesn't have any impact on the metrics used to assess transient faults, and might require significant additional analysis effort, it is possible to just choose the worst case and classify those faults as multi-point faults.

I. Memory element fault analysis

Faults occurring inside memory elements can stay on their outputs for an indefinite number of cycles. While trying to recalculate the impacts of such faults based on an existing regression run is possible, the scope of such recalculation is much larger than in the case of logical faults both in time (multiple cycles) and in cells impacted (not only the logic to the next sampling point, but also memory elements and logic beyond that), and might consume as much resources as actual rerunning. Instead of recalculation, we therefore propose to rerun a limited simulation, starting only from the cycle in which the fault was injected, and considering only the cells in the overlapping areas of the fan-out cone of the faulty memory elements and the fan-in cone of the safety critical logic.

Figure 3 below shows how the relevant area is calculated. In general it is easier to do this calculation on a gate level netlist, than on RTL. Assuming the fault is injected at the output of the left hand FF and the FF in the red circle is safety critical (output or critical SM), then only the green cells are relevant to the fault simulation. Any signal feeding into these cells from other logic could be red from the log file for the test, as it can't be impacted by the fault. The blue cells can be impacted by the fault, but are not within the fan-in cone of any safety critical logic, and hence don't need to be a part of the simulation.

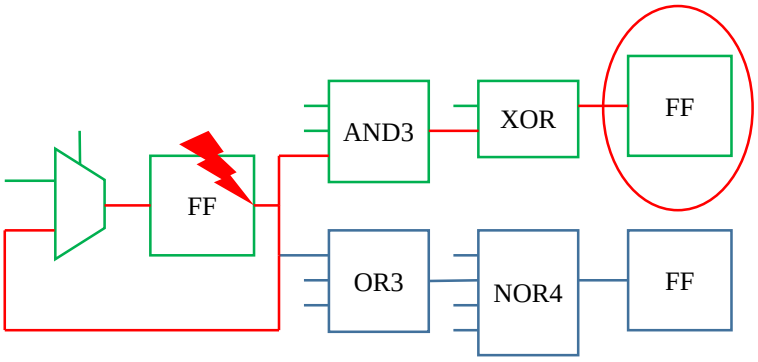

Figure 3: Bounding the scope for a simulation of fault in memory element

The bounded simulation rerun can finish in one of the options listed below:

1. Fault reaches safety critical logic
2. Fault disappears from system
3. Test terminates before 1 or 2 had happened.

Option #1 can be identified by continuously comparing the values of all safety critical signals against the reference run. If a difference is found, the fault has reached safety critical logic and is therefore classified as single point or residual fault. Note that this comparison should be continuous and not periodical, because the difference on the outputs might disappear.

Option #2 can be identified by periodically comparing the values of all signals in the bounded area against the reference run. Note that since only the momentarily value is being compared, this comparison is not too resource intensive. If no difference is found the fault has disappeared from the system and can be classified as "multi-point fault". As in the case of logical fault analysis, injecting X instead of a bit flip value can save the comparison against the reference run. In this case, the fault has disappeared if no X values are found in the system.

Option #3 requires the user to decide how the fault should be classified. This decision should be done based on areas which the fault has propagated to (which are, once again, easier to identify if X represents the fault). If it is estimated that the fault is only lacking some additional stimuli to reach safety critical areas, it should be classified as single point or residual. Usually, in case of uncertainty, the faults are classified according to the worst case, i.e. single point or residual.

J. Automation

The process described above could be automated to a large extent. The choice of N points for fault injection can be done by randomly choosing test, time and signal from the entire regression run. Once the N points are chosen the bounded design scope for each point can be identified, by calculating the intersection of the injection point fan-out, and safety critical area fan-in. This bounded design scope is then rerun and the results compared against the original test run, or checked for 'X'. All of these steps are easy to automate on most advanced verification platforms. The only case where manual intervention is required is option #3 in section I, where the fault is still present in the test by the time it ends. In this case, the user needs to decide how the fault is classified, or if a longer run or different test are required in order to make a decision.

IV. Summary

In this paper we have tried to address the challenge of transient fault analysis required by ISO 26262 and show how a practically infinite coverage space could be managed and analyzed with a user defined accuracy. This was achieved by a twofold approach including statistical sampling to represent the infinite space in with a finite number of tests and

a technique to accelerate individual fault analysis tests and reduce the resources they require, by reusing existing regression results.

In the future this work could be extended and improved in many ways. First, the same process could be repeated for other fault models with large or infinite coverage space such as stuck-at faults, or direct-current faults. While many parts could be shared, there are also important differences, as for these kinds of faults ISO 26262 also requires a distinction between latent and detected faults. Second, if the statistical model could be improved to include dynamic simulation data such as the percentage of time gates are in a given state, fault sampling could be improved, and the faults analyzed could also point to the areas that are the most sensitive to transient faults. Finally, integration of de-rating that could be done at higher abstraction levels into the overall FIT rate and metrics, could make those even more accurate and reduce overdesign.

V References

[1] ISO 26262 Road Vehicles – Functional Safety, Section 5-8
[2] ISO 26262 Road Vehicles – Functional Safety, Section 5-D
[3] ISO 26262 Road Vehicles – Functional Safety, Section 1 https://www.iso.org/obp/ui/#iso:std:iso:26262:-1:ed-1:v1:en:term:1.140
[4] ISO 26262 Road Vehicles – Functional Safety, Section 10-8.1.4
[5] Formula for standard error of sample estimates http://stattrek.com/estimation/standard-error.aspx?Tutorial=AP
[6] Formula for upper cumultive distribution function of normal distribution https://en.wikipedia.org/wiki/Normal_distribution
[7] Upper cumultive distribution values calculated by online calculator at http://keisan.casio.com/exec/system/1180573188
[8] ISO 26262 Road Vehicles – Functional Safety, Section 5-C.2
[9] ISO 26262 Road Vehicles – Functional Safety, Section 10-8.3.3
[10] ISO 26262 Road Vehicles – Functional Safety, Section 5-C.1.2
[11] "A Multi-Partner Soft Error Rate Analysis of an InfiniBand Host Channel Adapter", Chapman H., Landman E., Margarit-Illovich, A., Fang, Y.P., Oates, A.S., Alexandrescu, D. and Lauzeral, O., SELSE 2010.
[12] ISO 26262 Road Vehicles – Functional Safety, Section 10-A.3.2