Using an innovative SoC-level FMEA methodology to design in compliance with IEC61508

Riccardo Mariani, Gabriele Boschi, Federico Colucci

YOGITECH SpA
Pisa, Italy
http://www.yogitech.com

Abstract

This paper proposes an innovative methodology to perform and validate a Failure Mode and Effects Analysis (FMEA) at System-on-Chip (SoC) level. This is done in compliance with the IEC 61508, an international norm for the functional safety of electronic safety-related systems, of which an overview is given in the paper. The methodology is based on a theory to decompose a digital circuit in "sensible zones" and a tool that automatically extracts these sensible zones from the RTL description. It includes as well a spreadsheet to compute the metrics required by the IEC norm such Diagnostic Coverage and Safe Failure Fraction. The FMEA results are validated by using another tool suite including a fault injection environment. The paper explains how to take benefits of the information provided by such approach and as example it is described how the methodology has been applied to design memory sub-systems to be used in fault robust microcontrollers for automotive applications. This methodology has been approved by TÜV-SÜD as the flow to assess and validate the Safe Failure Fraction of a given SoC in adherence to IEC 61508

1. Introduction

New technologies allow deep system integration in automotive as well: replacing mechanics with electronics becomes to be a reality. Therefore, automotive System on Chip (SoC) are more and more complex: they have a mix of commodity and safety functions, an increased use of third-parties IPs and complex interconnection scenarios. On the other side, as a consequence of such increased complexity, the population of faults is increasing as well. These include: modelling uncertainty, functional verification holes, unforeseen interactions and misuse, specification misunderstanding, more electromagnetic susceptibility, soft-errors and malicious accesses. In particular, hardware faults (systematic or random) are worsened by: the increased soft-error failure rates (i.e. cosmic rays); coupling effects and disturbances are more and more important; and intrinsic uncertainty due to model inaccuracy is a problem of new technologies. If we define "robustness" as the ability to continue mission reliably despite the existence of systematic, random or malicious faults [1,2], how to make such systems more robust ?

For automotive, aerospace, biomedical and similar applications where the human life is concerned, safety is the driving factor. In such context, fault-oriented quality metrics (e.g. ppms) are not enough since they mostly confine the reliability issues to the semiconductor duty.

International norms exist to define requirements for safety, such the IEC61508 for functional safety of electrical/electronic/programmable electronic safety-related systems [3,4] or its "customization" to the automotive field, the ISO26262, still in the preliminary definition phase. Therefore, designers of electronic systems to be used in safety-critical applications should take into account these requirements and adapt their architectures. It is worth to note that the IEC61508 introduces also requirements in terms of design flows and validation criteria, so all the implementation process – from specs to verification and validation - should be adapted accordingly.

However, these norms generally refer to complete system and not to System-On-Chips: even if they contain also guidelines and requirements for the system components (including CPUs, memory systems, bus infrastructure and so on) and even if an extension of IEC61508 to ASIC is likely to appear in the next months, nevertheless it doesn't exist yet a consolidated methodology to systematically transport at SoC level the IEC61508 requirements. For instance, it's not so trivial to compute the Safe Failure Fraction (SFF, better defined in the next section) of a SoC and also the extension of system-level methods such Failure Mode and Effects Analysis (FMEA) to SoC is still confined to low-complexity integrated circuits or to basic critical points such muxing or digital-to-analog interfaces.

This paper shows how to make use of the FMEA at System-on-Chip (SoC) level as well and how to take benefits of the information provided by such analysis to implement a structured approach to increase the robustness of the SoC. This is done in compliance with the IEC61508, i.e. taking into account the failure modes and requirements therein described and it allows extracting the main metrics required by the norm. It will be also described how the proposed methodology has been applied to design memory sub-systems to be used

978-3-9810801-2-4/DATE07 © 2007 EDAA

in fault-robust microcontrollers for automotive applications.

2. IEC61508 basic concepts

The basic concept of IEC61508 is the definition of "Safety Integrity Level" (SIL), i.e. the discrete level (one out of a possible four) for specifying the safety integrity requirements of the safety functions to be allocated to the safety-related systems, where safety integrity level 4 has the highest level of safety integrity and safety integrity level 1 has the lowest [3]. As already said, the IEC61508 requirements generally are related to complete systems: however, also for system components it can be said that the safety integrity level is granted based on the value of Safe Failure Fraction (SFF) for the given component. SFF is equal to the ratio between the sum of safe failures (i.e. failures which don't have the potential to put the safety-related system in a hazardous or fail-to-function state) and detected dangerous failures over the sum of all the possible failures (safe plus dangerous).

Another important concept is the Hardware Fault Tolerance (HFT). A system with a HFT of N means that N+1 faults could cause a loss of the safety function. With a HFT equal to zero, a SFF equal or greater than 99% is required in order that the system or component can be granted with SIL3. With a HFT equal to one, the SFF should be greater than 90%. It is worth to be noted that SIL3 is the safety integrity level required for x-by-wire systems or systems with high criticality such active brake systems.

The IEC61508 also specifies faults or failures to be detected during operation or to be analyzed in the derivation of safe failure fraction: some examples are the following. For variable memories: DC fault model for data and addresses; dynamic cross-over for memory cells; no, wrong or multiple addressing; change of information caused by soft-errors. For processing units: DC fault model for data and Addresses for both internal registers and RAMs; Dynamic cross-over for memory cells; Wrong coding or wrong execution for coding execution including flag registers and so on.

The norm also assesses some of the state-of-art techniques for fault-detection and tolerance respect the maximum diagnostic coverage (i.e. probability of detection of dangerous failures) considered achievable: as example, RAM monitoring with Hamming code or ECCs or double RAMs with hardware/software comparison are the ones with the highest value.

The IEC61508 specifies as well which kind of documentation and design flow should be followed, such as the release of a Safety Requirements Specification (SRS) including a detailed FMEA (Failure Mode and Effects Analysis) of the system or sub-system.

3. Extending FMEA to SoC: the principles

The commonly used way to provide the information required by SRS is to perform a Failure Modes and Effects Analysis. This paper presents a way to perform the FMEA at SoC level with a systematic approach, supported by a spreadsheet and a tool to extract the information from the RTL.

In a first step, a set of "sensible zones" are identified from the RTL description. A sensible zone is one of the elementary failure points of the SoC in which one or more faults converge to lead a failure (Figure 1).

Valid definitions of sensible zones are:

- Memory elements such registers, flip-flops or variables. These sensible zones are points where many kinds of faults converge. Example: stuck-at or bridging faults in the combination logic generating the input of the memory element.
- Primary input and primary outputs of the SoC
- Logical entities that can or cannot directly map to a memory element. Example: wrong conditional field of a conditional instruction, where this wrong field can be caused either by a bit-flip or by a wrong processing of logic reading opcode from the bus.
- Critical nets such clocks or long nets that could generate multiple failures.
- Entire sub-blocks, to take more simply into account bigger cones of logic or to consider all together a complex block with a small number of outputs. Example: faults in a coder bringing to a wrong output value.

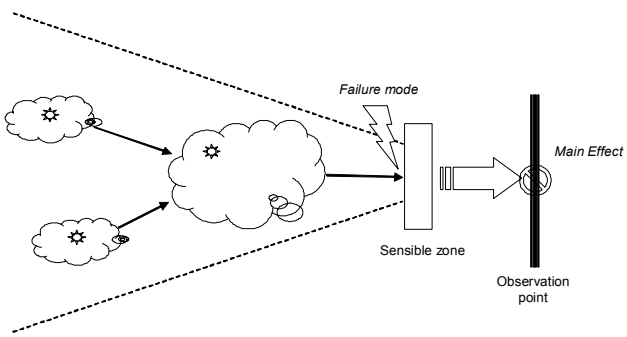

Figure 1: the sensible zone

It is worth noting that electronic circuits, in particular processing units, are mostly architected as groups of interconnected Moore machines. In such structures, the state register has a fundamental role in the functional behaviour of the machine, so it is worth to consider such state registers as the best candidates to become sensible zones.

Another important element of the SoC-level FMEA is the "Observation point". The observation point is either: another sensible zone, a primary output (most of the cases), a primary function of the SoC (when the analysis is more high-level) or an alarm of the diagnostic. The effects of failure modes in a sensible zone are measured at these observation points.

Failure modes can be of two main types. It can be directly linked to physical faults. Example: if the sensible zone is a memory element, it can be a bit-flip in the register. It can be the end consequence of faults in the logic cone of the sensible zone. Example: a wrong value in a register bit due to stuck-at or bridging faults of the combinational logic in front of the D pin of a register. Failure modes can be also a temporal sum of faulty events (such multiple faults hitting a memory element). The basic failure modes for a given SoC can be determined from the tables in Appendix of IEC 61508-2 [3].

Concerning the correspondence between failure modes of sensible zones and HW faults of their converging cones, it is useful to distinguish three classes of physical HW faults: local, wide and global HW faults.

We consider "local" the physical HW faults affecting one or more gates of a logic cone contributing to a single sensible zone. Each local HW fault or combination of them occurring in the logic cone in front of the sensible zone – if not masked by conditions or by other HW faults - will result in a failure in it.It is worth to note that if a certain local HW fault is masked so it doesn't generate any effect in the sensible zone (e.g. if there is a transient fault in a gate but this glitch isn't sampled by the clock of the register corresponding to its sensible zone and so on), this fault is not considered as an hazard since it doesn't perturb the function to be performed by such sensible zone. The type of failure that will occur in the sensible zone depends on the type of occurred physical faults (e.g stuck-at, bridging fault, etc…).

We consider "wide" the physical HW faults affecting one or more gates of a logic cone contributing to more than one sensible zone. Examples of wide physical HW faults is a single physical HW fault (e.g. a stuck-at at the output of a gate) generating a failure in two or more sensible zones, or a single physical HW fault belonging to a logic cone contributing to two or more sensible zones but generating a fault only in some of these zones (see Figure 2). In such a case, we have multiple failures. It is worth to note that such case also includes situations like faults in clock or reset buffers affecting multiple flip-flops. Physical faults like resistive or capacitive coupling between lines are also included in such model.

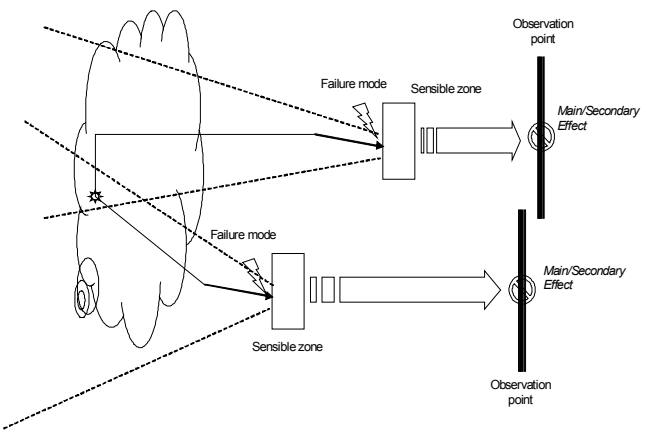

Figure 2: multiple failures

We consider "global" the physical HW faults affecting many logic cones and therefore contributing to more than one sensible zone. Examples of global physical HW faults are the following: faults in the PLL or clock generation or first level of clock trees affecting large number of sensible zones; power supply faults affecting large areas of the silicon component; thermal faults making slower consistent region of the SoC.

Concerning the effects of a fault, we define the "main effect" as the effect that at least will occur as result of failure mode of the considered sensible zone respect an observation point, if not masked internally. The "secondary effects" are the other effects occurring at other observation points resulting from the migration of the sensible zone failure through its output logic cone and from there to other sensible zones till the other observation points. These are particular important to take into account the very frequent situation in which a single local HW fault generates a failure of a single sensible zone, but the effect manifests itself at different observation points (see Figure 3).

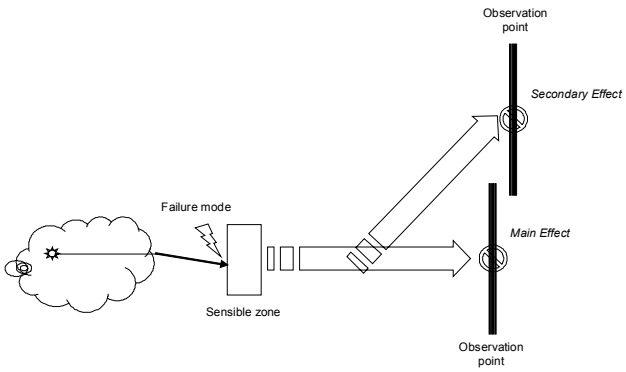

Figure 3: secondary effects

The extraction of sensible zones and observation points is automatically performed by a tool based on commercially available EDA tools such Cadence or Synopsys, working on the synthesized RTL. Besides to collect and properly compact the registers, the tool extracts as well the data needed by the FMEA statistical model, such the composition of the logic cone in front of each sensible zone (i.e. gate-count, interconnections and so forth) and the correlation between each sensible zone in terms of shared gates and nets.

Starting from the elementary failure in time (FIT) per gate and per register both for transient and permanent faults, all the data automatically extracted by the tool are used to compute the failure rates for each sensible zone. A spreadsheet contains all these data as also other information provided by the user, such:

- S and D factors to estimate the Safe fraction and Dangerous fraction of the possible failures for the given failure mode in the given sensible zone. Two types of S and D factors are used: architectural and applicational. Example of architectural dependent S/D is a sensible zone always inactive at run-time because blocked by a set of masking gates. Example of application dependent S/D is a sensible zone not used by the given application. Usually only architectural S/D factors are considered.
- The frequency class F of the given sensible zone, used to estimate its usage frequencies.
- The lifetime ζ, defined as the time between the average last read and the write in such zone.

Based on this information, the spreadsheet computes all the metrics required by the IEC61508, such as the safe (λ_S) and dangerous (λ_D) failure rates for each sensible zone and for all the SoC. It also delivers a ranking of sensible zones in terms of their criticality.

The proposed approach is therefore a mix between analysis performed at different levels, such RTL level (for the estimation of S, D and F factors) and at gate-level (for the statistics related to the logic cones of the sensible zones and so forth): this way guarantees the best accuracy of the results and offers as well the possibility

to analyze which of different possible implementations is the more critical in term of safety.

4. Using the FMEA to design diagnostic

The methodology proposed in this paper has the specific target to evaluate the SoC in order to find the best strategy for error detection and correction. Therefore, the two main quantities that have to be measured are the Diagnostic Coverage and the Safe Failure Fraction, defined by the following formulas [3]:

$$DC = \frac{\sum \lambda_{DD}}{\sum \lambda_{DD} + \sum \lambda_{DU}} \qquad SFF = \frac{\sum \lambda_{S} + \sum \lambda_{DD}}{\sum \lambda_{S} + \sum \lambda_{DD} + \sum \lambda_{DU}}$$

where λ_{DD} is the rate of dangerous detected failures and λ_{DU} is the rate of dangerous undetected failures ($\lambda_D=\lambda_{DD}+\lambda_{DU}$).

To compute such values, the spreadsheet includes for each sensible zone the fraction of the dangerous failure rate associated with each failure mode that is claimed to be detected by the diagnostic technique, distinguished in Detected Dangerous Failure fraction (DDF) for transient/intermittent faults and permanent faults. It is also distinguished between DDF due to HW and SW techniques.

These coverage values are computed both based on the architecture, by the numbers given by the previous described tool (concerning the interconnections between sensible zones), by what accepted by the IEC norm (Annex 2, tables A.2-A.13, where it is specified the maximum diagnostic coverage considered achievable by a given technique) and by the estimation of the user. The FMEA validation flow described in next section must be executed in order to have the highest level of confidence in such estimations.

An important step of the FMEA is to span the values of the assumptions (such the elementary failure rates for transient and permanent faults or the user assumptions such S, D and F) in order to measure the sensitivity of the final DC/SFF to these changes.

5. How to validate the FMEA

A strict and measurable validation flow is rather important in order to cross check the FMEA. As recommended by the IEC61508 norm, fault injection has a crucial role in that. The proposed methodology uses a validation flow based on a mix of tools which the main ones are a simulation-based fault injector [8,9] and a fault simulator like [11]. The fault injector tool is built on top of a state-of-art functional verification tool [10] and makes use of a standard verification language [12]. By integrating fault injection with functional verification, it is possible to set up a fault injection flow that solves many of the issues that affect most of the environments presented in literature. Thanks to the interaction with the functional verification tool, verification components available on the market can be easily reused as a workload to inject faults, obtaining at same time design validation and reliability evaluation. The use of a standard language enables an easy and configurable way to model the faults. The engine of the coverage-driven functional verification tool allows to uniquely correlating Workload, Operational Profiles, Fault List, and final measures.

The fault injector is composed by (figure 4):

- *Environment builder*: this block extracts from the FMEA all the information related to the environment for the injection campaign and builds all the required environment configuration files.
- *Operational Profiler, Collapser and Randomiser*: starting from the information extracted by the Environment Builder, this block extracts the Operational Profile (OP) from a given workload. An Operational Profile (OP) is a collection of information about all relevant fault-free system activities: traced information items are read/write activity associated with processor registers, address bus, data bus, and memory locations in the system under test, but they may also include other more high level information like the most probable expected sets of inputs that the system or application will receive. The purpose of the OP is to better understand the situation in which the system or the application will be used, and then analyze this information to ensure that only faults which will produce an error are selected during the fault list generation process. In this way the generated fault list is compacted and non trivial. As mentioned afterwards, the completeness of the workload is measured in a deterministic way to check if it complete in terms of its capability to trigger all the sensible zones of the DUT.
- *Fault Injection Manager:* this function runs all the injection campaign based on automatically generated fault lists and collects all the results.
- *Result analyzer:* this function collects all the results generated by the injection campaign and automatically fills a sheet included in the FMEA spreadsheet. In particular, S, D, F and DDF are extracted and compared with the values in the FMEA. The validation is successful if the percentages are in line with the estimated values. It is also extracted a "table of effects" for each sensible zone, i.e. table of observation points in which has been measured a deviation respect a golden simulation without injected faults. This table is automatically compared with the FMEA to check if the identification of main/secondary effects is consistent.
- *Monitors and Coverage Collection:* this function, composed by a set of monitors automatically instantiated by the environment builder, generates and collects all the information needed to build the coverage measures for the analysis of fault injection campaign completeness. In this context, "coverage" means a measure of the completeness of the fault injection experiment. It is measured how many times a fault injection point (SENS) is triggered by an injection, how many changes occurred on the observation point (OBSE), how many mismatches occurred between faulty and golden DUT, how many times the diagnostic point (DIAG) changed and so forth. Only when all the

coverage items are covered at 100% we can consider complete the fault injection experiment.

The validation procedure is the following:

a) it is performed an exhaustive fault injection of sensible zone failures: based on the failure mode and condition specified in the FMEA and on the workload and extraction of operational profile, for each sensible zone it is injected a certain number of faults. At the end of this analysis, both the results and the coverage are cross-checked with FMEA.

b) in parallel, the efficiency of the workload in covering the HW gates of the gate-level netlist is measured, for instance by using a toggle count coverage or a standard fault coverage. If the toggle count percentage (i.e. nets/gates toggling at least once) or the fault coverage is greater than a defined value (default 99%), the validation is successful.

c) for critical areas (where the analysis is more difficult) or where particular HW implementation are present (asynchronous circuitry and so on), a selective HW fault injection is performed, injecting local faults with fault injector. The validation is successful if the results of such injection confirm the results of the exhaustive sensible zone failure fault injection. Otherwise, some new lines should be included in the FMEA to take into account the newly detected effects. For these critical areas, the fault simulator can be used to precisely measure the fault coverage vs permanent faults respect the workload and the implemented diagnostic. The validation is successful if the results of such run are in line with DDF estimated in the FMEA sheet

d) for wide/global HW faults, a selective fault injection is performed. The validation is successful if the results of such injection confirm the results of the exhaustive sensible zone failure fault injection. Otherwise, some new lines should be included in the FMEA to take into account the newly detected effects.

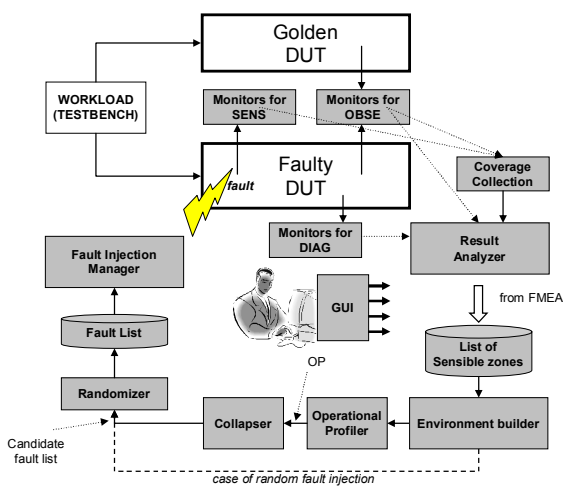

Figure 4 : the fault injector

6. Example

To show how this methodology can be successfully applied to the design of safety-critical SoCs, a proof-of-concept example is described in the following. It is the

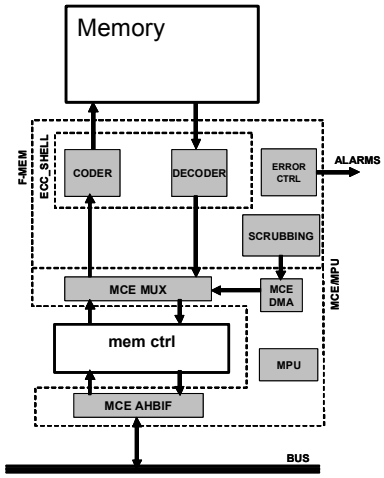

Figure 5: the memory sub-system

design of a memory sub-system, based on the architecture already presented in [6,9]. This architecture is represented in figure 5 and it is mainly composed, besides the memory controller and the memory array, by a memory protection IP composed by two different functional units:

a) F-MEM: it interfaces the memory array and it hosts the coder/decoder and a "scrubbing" feature, as also the controller to generate the corresponding alarms. In a few words, The scrubbing function stores the locations where an error occurred, in order to repair them when the memory isn't used by the system or it can also perform a background scanning of the memory for fault-forecasting.

b) MCE: it interface the F-MEM with the memory controller and with the bus, providing the DMA access for F-MEM scrubbing feature as also a "distributed MPU" functionality. This MPU function considers that the memory is divided in number of pages associated with attributes and permissions. The MCE block uses signals from the bus (in such a case a AHB multilayer bus) to discriminate these attributes and permissions and in case of faults, proper alarms are generated.

The FMEA methodology has been applied to such architecture with the goal of achieving a SIL3 memory sub-system, i.e. with a SFF equal or greater than 99%.

A first implementation of the memory sub-system was done. Concerning the coder/decoder, a SEC-DED algorithm was used with a standard modified Hamming architecture. This first circuit included a write buffer and a pipeline stage in the decoder, in order to guarantee the timing closure and to avoid the degradation of the memory access time due to the ECC.

At first, the sensible zones have been extracted by using the previously described tool: about 170 sensible zones resulted, including the memory controller, the memory and the F-MEM/MCE blocks. The memory has been modeled by using a proper fault model as for instance described in [13-15]. Then, the FMEA spreadsheet have been completed including S,D, F and DDF values following the procedure described in the sections 3 and 4.

The spreadsheet identified the critical zones. Besides the memory array itself, the most critical blocks were the BIST control logic, the registers involved in addresses

latching, most of the blocks of the decoder, the registers of the write buffer, some of the blocks of the MCE handling the interconnections with the bus and so forth. With the initial implementation, resulting SFF (around 95%) was not enough to reach SIL3. Then, the architecture was modified by adding the addresses to the coding (required as well by IEC61508), by adding parity bits to the write buffer and by deeply modifying the decoder implementation. In particular, this last action was really important to increase the SFF:

i) an "error checker" was added immediately after the "code generator" section of the decoder, in order to cover also the errors in such coder;

ii) a double-redundant "error checker" was implemented after the intermediate decoder pipeline stage, to check the correctness of code and data fields after the pipeline as also – in case of no errors – directly connect the decoder output with the memory data. The spreadsheet shown that this measure was strongly decreasing the error probability of the second part of the decoder architecture;

iii) a "distributed" syndrome checking architecture was implemented to allow a finer error detection (i.e. to discriminate if an error is in the code field, or in data field or if it was an addressing error, etc…). As shown in the FMEA, also this architecture strongly decreased the error probability. New alarms were generated by these checking architectures: as shown by the FMEA, by combining the alarms generated by the error checker after the decoder's coder, the redundant error checkers after the pipeline and the final syndrome checks, it is possible to cover with a very high level of coverage the possible error combinations in the decoder.

Moreover, some SW start-up tests were identified for the memory controller parts not covered by the memory protection IP. The resulting SFF of this second implementation was 99,38% and it was very stable as well, i.e. changes on S,D,F and fault models didn't change the result in a sensible way. The previous described validation flow was run in order to have the highest confidence on the results, with different synthesis of the design in order to cross check the sensitivity to the final implementation.

7. Conclusions

In summary, the methodology proposed in this paper is a new way to extract useful information from a SoC, to take into consideration the IEC guidelines about fault models and failure modes, to compute (following IEC 61508 norm) the Safe Failure Fraction and the Diagnostic Coverage, to validate the results by means of a complete flow including a fault-injector. It's an innovative and systematic approach to assess the safety of a circuit, delivering very detailed reports on sensible zones, fault effects, failure rates, etc… that can be used for SoC analysis. It allows the identification of critical part of a circuit and the exploration of possible implementations for best safety as well.

The methodology has been developed under the supervision of TÜV-SÜD and it has been approved by TÜV as the flow to assess and validate the Safe Failure Fraction of a given SoC in adherence to IEC 61508.

The methodology has been used to certify the fRMEM product of YOGITECH SpA according IEC 61508. It is currently in use for the final certification of the other IPs of YOGITECH faultRobust technology and for the complete analysis of fault-robust microcontrollers for automotive applications [16,17].

References

[1] J.C Laprie, "Dependable Computing and Fault Tolerance Concepts and Terminology ", IEEE Computer, 1985

[2] H. Tahne, "Safe and Reliable Computer Control: Systems Concepts and Methods", Mech. Lab, Univ. Stock, 1996

[3] CEI International Standard IEC 61508, 1998-2000

[4] S.Brown, "Overview of IEC 61508 Design of electrical/electronic/programmable electronic safetyrelated systems", Computing & Control Engineering Journal February 2000, pages 6-12

[5] R.E. McDermott et al, "The Basic of FMEA", Quality Resources Press, 1996

[6] R. Mariani, G. Boschi, "A System Level Approach for Embedded Memory Robustness" Special Issue: Papers selected from the 1st International Conference on Memory Technology and Design - ICMTD'05.

[7] R. Mariani, M. Chiavacci, S. Motto, "Dependable microcontroller, method for designing a dependable microcontroller and computer program product therefor", European Patent, EP1496435

[8] R. Mariani, P. Fuhrmann, B. Vittorelli, "Cost-effective Approach to Error Detection for an Embedded Automotive Platform", SAE 2006 World Congress & Exhibition, April 2006, Detroit, MI, USA

[9] www.fr.yogitech.com

[10] http://www.cadence.com/products/functional_ver

[11] http://www.cadence.com/products/digital_ic/encountertest

[12] IEEE standard 1647, http://www.ieee1647.org/

[13] S. Mukherjee et al. "Cache scrubbing in Microprocessors: Mith or Necessaity?", 2004

[14] S. Mukherjee et al. "A Systematic Methodology to Compute the Architectural Vulnerability Factors for a High-Performance Microprocessor", 2003

[15] M. Spica, "Do we need anything more than single bit error correction (ECC)?", 2004

[16] R. Mariani, "A Platform-based Technology For Fault-robust Soc Design", IP/SOC 2006 Conference, December 2006, Grenoble, France

[17] R. Mariani, P. Fuhrmann, B. Vittorelli, "Fault-Robust microcontrollers for automotive applications", 12th IEEE International On-Line Testing Symposium - 12 July 2006 - Como,Italy