WHITE PAPER

Srikanth Rengarajan

VP – Products & Business Development

Case Studies in Hardware Functional Safety Analysis

(*Or* "Why Spreadsheet don't work for FuSa Architects")

Addressing Functional Safety in Automotive electronics has historically been a co-operative endeavor between the system designers from the Tier1/OEM space and the component suppliers. The effort predates the now-dominant ISO26262 standards and the associated ASIL classifications and relied on system(ECU)-level Failure Modes and Effects Analyses (FMEA) to lay the groundwork for component failure rates and diagnostic metrics. While the arrival of ISO26262 and, in particular, the part11 of ISO26262-2018 have formalized the target metrics and the definitions of coverages, the instruments for safety analysis still remain largely historical. For the most part, estimates of FITs and DC are made at the IC or IP level based on previous iterations, industry rules-of-thumb or prior-art and fed into a spreadsheet based calculation. This article analyzes the pitfalls of such an approach and argues for a robust tool-driven analytical technique based on real-world customer data.

Automotive Safety Analysis Flow

FMEA, and its deductive counterpart FTA (Fault Tree Analysis) lay the ground-work for functional safety analysis in any design endeavor. The results of this analysis are recursively applied to each level of the system design starting at the ECU level down to the elementary sub-part, typically an electronic or electromechanical component. The ISO26262 standard provides exhaustive guidelines on classifying the failure potential of these elements within a rigorous framework of systematic and random faults, resulting in functional safety integrity levels(ASILs). Given a target classification, the electronic designer carries the burden of proposing remedial measures which together lead to a FMEDA (Failure modes and Effects Diagnostic analysis). In popular parlance, these remedial measures are termed Safety Mechanisms.

Figure 1 shows an example calculation of the probabilistic random hardware failure coverage metrics for a CPU. The designer/architect analyzes the design at a chosen level of abstraction (here at the constituent functional units of the CPU) and proceeds to derive the safety metrics. Broadly this entails the following steps:

1. *Identification of Safety Elements*: For each sub-part of the design, an estimate is made of the safety-criticality. Non-safety-related items are excluded from the calculations.
2. *Failure mode mapping:* The FMEA results provide various failure modes of the safety element. The designer has to decompose these modes into the sub-parts under consideration and map them to Fault models. As an example, a failure mode of the CPU could be "Wrong data output".

This is mapped on to the ALU as “wrong data generation by the addition unit” which is then mapped to “a stuck-at-1 fault on the carry-input of the adder”.

3. *Failure Rate distribution:* The failure rates of each sub-part by fault model needs to be entered. The estimation of these failure rate distributions is an exercise in trial and error based on past experience and industry data. Given the multitude of design element within each sub-part and the number of fault models that could apply to each, virtually all designers choose to stay at a sub-part level of granularity for their analyses. In the example of Figure 1, the fault model is simply “Permanent” and “Transient” faults at the level of ALU.
4. *Safety Mechanism Coverage Estimation:* The designer estimates the diagnostic coverage provided by any safety mechanism present in the design unit. For the most part, these coverage numbers are sourced from the ISO26262-2011:5 guidelines from Annexe-D, for all common mechanisms. For custom safety mechanisms, particularly proprietary system level constructs, the designer carries the burden of accurate estimation of coverage.

					Permanent failures								Transient failures				
Part	Sub-part		Safety Related Component ? No Safety Related Component ?	Failure modes	Failure rate (FIT)	Amount of safe faults (see note 1)	Safety mechanism(s) preventing the violation of the safety goal	Failure mode coverage wrt. violation of safety goal	Residual or Single Point Fault failure rate / FIT	Safety mechanism(s) preventing latent faults	Failure mode coverage wrt. Latent failures	Latent Multiple Point Fault failure rate / FIT	Failure rate (FIT)	Amount of safe faults (see note 1)	Safety mechanism(s) preventing the violation of the safety goal	Failure mode coverage wrt. violation of safety goal	Residual or Single Point Fault failure rate / FIT
CPU	Register bank	Register R0	SR	permanent fault	0.0029	0%	SM1	40%	0.00174	SM1	100%	0.00000					
				transient fault									0.032005	0%	SM1	40%	0.01920
		Register R1	SR	permanent fault	0.0029	0%	SM1	40%	0.00174	SM1	100%	0.00000					
				transient fault									0.032005	0%	SM1	40%	0.01920
		Register R2	SR	permanent fault	0.0029	0%	SM1	20%	0.00232	SM1	100%	0.00000					
				transient fault									0.032005	0%	SM1	10%	0.02880
		Register R3	SR	permanent fault	0.0029	0%	SM1	20%	0.00232	SM1	100%	0.00000					
				transient fault									0.032005	0%	SM1	10%	0.02880
	ALU	ALU	SR	permanent fault	0.0348	0%	SM1	20%	0.02784	SM1	100%	0.00000					
				transient fault									0.00038	20%	SM1	10%	0.00027
		MUL	SR	permanent fault	0.0290	0%	SM1	20%	0.02320	SM1	100%	0.00000					
				transient fault									0.00037	70%	SM1	10%	0.00010
		DIV	SR	permanent fault	0.0232	0%	SM1	20%	0.01856	SM1	100%	0.00000					
				transient fault									0.00036	70%	SM1	10%	0.00010
		Pipeline	SR	permanent fault	0.0174	0%	SM1	90%	0.00174	SM1	100%	0.00000					
				transient fault									0.00103	20%	SM1	90%	0.00008

Figure 1: ISO26262-11:2018: Annexe E. An IC-level Safety Metric Calculation example

Specific to hardware, part 5 of the standard details the Failure rates (FITs) and diagnostic coverage (DC) metrics. The same section also provides guidelines on estimating these metrics for a variety of common design elements and safety mechanisms.

As an example, a common mechanism is to add a parity bit to a sequence of binary valued registers in the digital logic. Any corruption of the design by a single bit would lead to an inversion of polarity and would be detected. The ISO standard, taking into account common implementation practice recommends that EDC mechanisms, as applied to memory structures receive HIGH(99%) coverage while similar mechanisms applied to bus structures receive MEDIUM (90%) diagnostic coverage. Most IC

Memory monitoring using error-detection-correction codes (EDC-ECC)	5.1.13.1	High	The effectiveness depends on the number of redundant bits. Can be used to correct errors

Figure 2: DC of memory element with EDC/ECC

designers who rely on literal transcription of the standard, deploy one or the other of these numbers as

part of their spreadsheet to arrive at the overall DC metric for their IC. However, these guidelines are often a poor match for reality for the following reasons.

Problems with standardized guidelines

Sequential vs Combinatorial Logic: Error-Detection circuits of the kind referenced by the standard (Eg. Hamming code) capture registers or state elements very well. However, the logic cone feeding into them is not covered. While the 99% coverage is appropriate for a memory array, the overall DC for the element or sub-part is heavily influenced by the proportion of combinatorial logic affecting the state element. The standard attempts to account for this by providing a column in the FMEDA tables (Annexe D, Part 5) titled "Failure Mode Coverage wrt violation of Safety Goal" which is an estimate of the proportion of contributing logic that the mechanism covers. But estimating that for a unit such as ALU or DMA is a non-trivial exercise and requires access to netlist-level design information which is often unavailable until late in the chip cycle.

Memory Elements: A variant of the above issue occurs when the level of abstraction of the analysis encompasses memory macros. Both FIT and DC are different for memory bitcells versus logic and the safety mechanisms affecting them are explicitly called out differently in the standard. Again, assigning a single DC number to a Multiply-Accumulate unit with multiple FIFOs, memory macros etc. is misleading. A correct answer is to analyze the part (MAC) by decomposing it into its sub-parts (multiplier, adder, register bank, memory, FIFO, debug, clocks…) and recomputing the random hardware failure metrics.

All nodes are not equal: FITs are typically characterized by transistor and/or area counts in the standard. Hence a elementary sub-part that is covered by a safety mechanism gets credit in the base calculations only to the extent of its contribution to the number of transistors it contains. Reality in a logic circuit is somewhat different. A Flip-Flop containing a mode-bit for ADAS computation enablement is clearly more crucial to safety than one that enables a debug trace pathway for software development. Some of this information can be gleaned from architectural guidance but a pure structural understanding by a synthesis-like tool can provide tremendous insight into the relative importance of elements of a design

Synthesis Effects: A true evaluation of Functional Safety metrics can only be accomplished at the gate level as the standard makes clear . Different synthesis options can result in corresponding variations in terms of skewing the DC sensitivity of the design. Higher level constructs like choosing the configuration of a memory macro or swapping-HVT transistors to save power can induce tremendous variations in the design's resilience. Capturing this at a manual level implies a tight hand-off between the safety architect and the back-end team in a design environment which is often impractical.

Control vs Data structures: A common concept in providing resilient on-chip transport is end-to-end safety protection (E2E). Typically, this takes the form of hardware redundancy at a single-bit(parity) or multi-bit level (ECC/EDC) which protects the address, control and payload information that is transported in a packet/transaction. This is a robust mechanism and the ISO mechanism allocates MEDIUM (90%) coverage for implementations supporting E2E-ECC. However, a closer analysis reveals that such mechanisms protect the packet only as long as the data is used as a single payload. In a standard network-on-chip architecture, portions of the address and control segments are needed for purposes of determining routing, QoS, arbitration etc. As soon as bits are peeled off the packet, the E2E protection is lost. In modern NoCs as present in ADAS SoCs, the proportion of control logic can be

moderate to severe. The standard's guidelines are a poor substitute for a tool-driven analysis in such cases.

A tool-driven approach

Safety practitioners in the electronic design industry have long recognized the need for a methodology that goes down to the elementary sub-part to reconstitute the design and produce up-to-date DC and FIT metrics. Clearly a tabular or spreadsheet method is inadequate to capture the depth of detail that makes for viable safety assessments. Austemper's ***SafetyScope*** tool was designed to address such a need and specifically takes into account the variables outlined earlier in a transparent and automated manner.

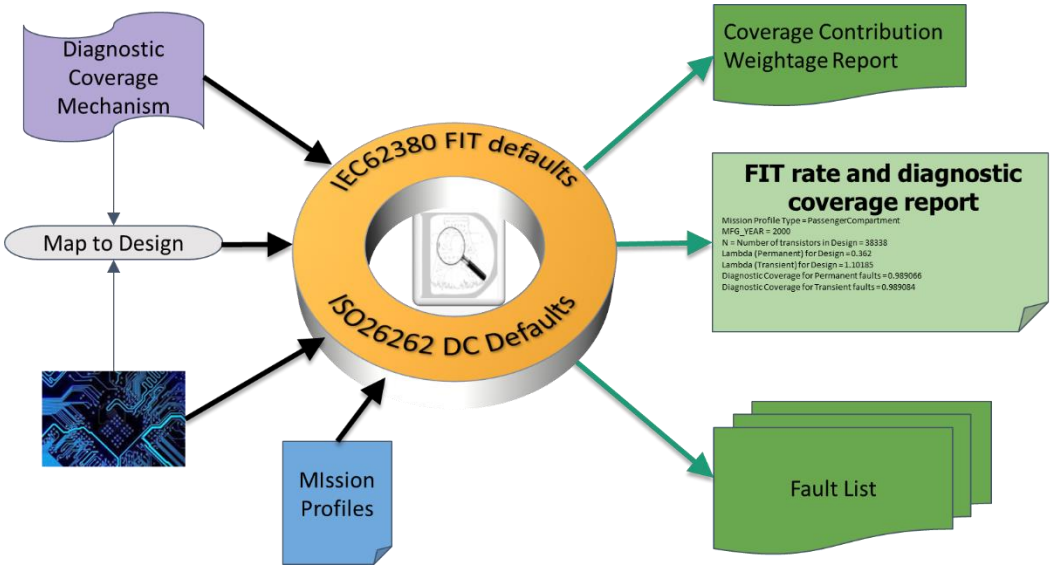

Figure 3: Austemper SafetyScope Functional Safety Analysis Tool

Case Study A: Global Automotive IC vendor – Peripheral Subsystem

One of our customers, a leading global automotive IC vendor, chose to deploy Austemper's ***SafetyScope*** tool on one of their internal functional blocks. The IP (Srik) comprised a Peripheral controller talking to a bus client over a shared ARM AXI crossbar. Of note is the fact that the bus client, manipulates the data prior to writing it into an internal FIFO. The significance of this operation lies in the context of the end-to-end protection mechanisms adopted by this IP for transporting data over the crossbar fabric. Specifically, the client had chosen to use a single-bit parity protection.

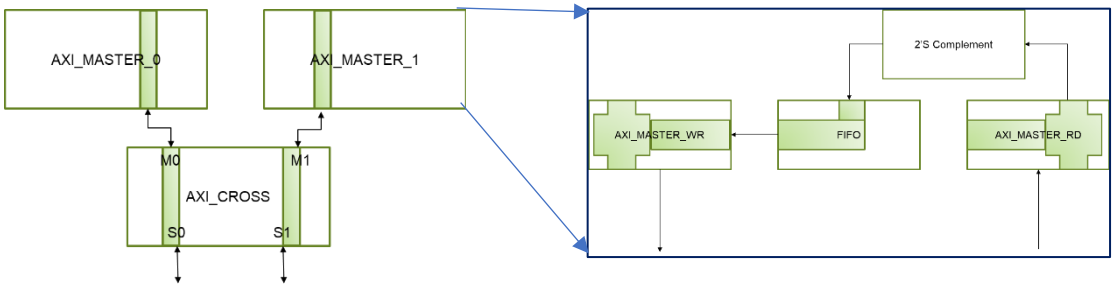

Figure 4: Client IP Abstraction - with detail view of Bus master

The analyzed fault models were simple stuck-at_1 and stuck-at_0. As can be expected, the client's internal analysis had projected a DC-permanent of 60% consistent with the recommendations of the standard for on-chip communication.

Table D.14 — On-chip communication

Safety mechanism/measure	See overview of techniques	Typical diagnostic coverage considered achievable	Notes
One-bit hardware redundancy	D.2.7.1	Low	—

Figure 5: ISO26262:2011-5: Annexe D

In other words, the standard recommends the user to claim **a 60% DC credit** for employing E2E parity.

For the express purpose of analyzing E2E safety constructs in bus structures, ***SafetyScope*** supports the **RIVERFLOW** mode of operation that automatically traces the logic inside the bus and assigns credit. It also identifies and separates control structures from datapath and assigns diagnostic credit only to covered sections of the fabric. Running the customer's design via ***SafetyScope's*** **RIVERFLOW** mode produced the following coverage

DIAGNOSTIC COVERAGE	VALUE
Permanent	17.7%
Transient	16.3%

Figure 6: SafetyScope Derived DC values

Analysis

Clearly, the initial estimates were considerably optimistic than the results of the structural analysis. Upon closer inspection, two reasons could be discerned for the discrepancy.

Control Structures

The E2E coverage was limited to the data portion of the design. A significant portion of the bus fabric was devoted to address-based routing, arbitration-logic and QoS, none of which is covered by the E2E safety mechanism, since they are based on a subset if the payload. While it is theoretically possible to extend this coverage by performing a parity check on every dataword that encompasses the control information, doing so is costly in terms of timing and area and typically not done. End-to-End Parity almost always implies checking the parity at the ends.

Data path termination

The presence of the 2's complement operation nullifies the impact of E2E for all logic beyond that point. Depending on the amount of such logic, and there was significant amount in the case of this customer's IP, the coverage levels varied drastically from the initial estimates.

Lesser factors in this scenario included configuration registers and architectural estimations of non-safety-critical logic which accounted for about 15% of coverage.

The result was an optimistic discrepancy of 35% of DC which had gone unnoticed in past incarnations of this IP and would likely have met the same fate but for the usage of *SafetyScope*.

Case-Study B: North American Automotive Startup – ADAS SoC

For our second case-study, we turn to another of our customers developing ADAS System-on-Chips and seek in-context certification for Radar safety applications. The ARM-CPU based SoC is designed to provide Radar sensor coverage for autonomous and semi-autonomous driving applications. As part of its toolkit, the SoC employs custom signal processing elements which are supported by the usual array fof peripherals and infrastructure IP elements to support the acquisition and analysis of the sensor signals. The overall system safety target is ASIL-D . FMEDA and ASIL decomposition resulted in the SoC being designated as ASIL-C with a safety island within being the only ASIL-D element on-chip. For our case-study we focus on three standard but safety-critical items within the SoC.

1. DMA Controller for managing the myriad peripherals on the platform.
2. An interrupt controller from a leading IP vendor for supporting the control software operation on CPU
3. A DRAM memory controller

As part of their functional safety architecture, the system designers chose to protect all the critical state elements within these IP-blocks using single-bit parity. Again, we turn to the standard (ISO26262-5:2011) for guidance on single bit hardware redundancy. While silent on coverage for standard combinatorial and Sequential logic (Table D.13), the recommended coverage for volatile memory serves as a substitute (Figure 7)

Table D.6 — Volatile memory

Safety mechanism/measure	See overview of techniques	Typical diagnostic coverage considered achievable	Notes
RAM pattern test	D.2.5.1	Medium	High coverage for stuck-at failures. No coverage for linked failures. Can be appropriate to run under interrupt protection
RAM March test	D.2.5.3	High	Depends on the write read order for linked cell coverage. Test generally not appropriate for run time
Parity bit	D.2.5.2	Low	—

Figure 7: Recommended coverage for Parity Insertion

At first glance, a **60% DC target** is a reasonable goal for the overhead involved in generating and checking a single bit parity. Again, Austemper's ***SafetyScope*** tool was used, this time in its baseline configuration as suitable for any IP block. The tool results for each of them is shown below

BLOCK	Flop-count	Gate Instances	State-element by area	Safety Mechanism	Target ASIL (DC)	SafetyScope DC Results	
						Permanent	Transient
DMA Controller	44,730	321,594	62.1%	Parity – All Registers	B (60%)	37.3%	98.5%
Interrupt Controller	8,068	286,053	23.9%	Parity – All Registers	B (60%)	59.5%	98.5%

Memory Controller	55,591	440,452	50.2%	Parity – All Registers	B (60%)	69.1%	98.8%

Figure 8: IP data and SafetyScope Results

Analysis

The results show an interesting pattern. On the permanent DC, only the Interrupt controller is tracking to target while the other two blocks are on either side. This is despite the designers inserting identical safety mechanisms across each design in its entirety. On the Transient front, all three IPs exceed their target and are close to ASIL-D levels. We look at each metric's derivation in depth below.

Permanent Fault Coverage

Taking the DMA controller first, we note that the proportion of state elements in the design – 62% - is actually quite close to the standard-recommended value. Hence adding parity to this portion of the design should validate the recommendation. The discrepancy arises from the failure to consider the structural implications of the design.

As mentioned in the preceding section, the ratio of sequential vs combinatorial logic is a key determinant in the diagnostic coverage metric. Specifically,

- Designs that exhibit high fan-in from primary inputs or register outputs would show greater resilience when those points are protected. (*State-Dominated)*
- Conversely, designs with large cones of logic feeding into state elements inherently lower the value of parity protection. (*Cone-Dominated*)

The DMA controller happens to be highly cone dominated. And in this case, the protection mechanism applies exclusively to state elements. Taken together, the resulting the coverage of permanent faults drops from 62% to 37%.

We can extend this analysis to the other two blocks. For the interrupt controller, the state-element ratio (and hence the portion of logic protected by parity) is actually very close to 60% and is deemed neutral. Finally, the memory controller displays a very high (relatively) proportion of state (state-dominated), there-by multiplying the power of parity protection. With the benefit of the structural analysis, ***SafetyScope*** deduces these proportions and apply the appropriate ratios towards diagnostic coverage.

Transient Fault Coverage

Transient fault, on the other hand, is uniformly high for all the blocks regardless of their structural connectivity. Why would this be so? By nature of synchronous design, any transient fault in a state element has a disproportionate impact on the design output relative to combinatorial elements, due to the potential to disturb stored state. A combinatorial element has much greater immunity to transients since the window of sensitivity is directly proportional to the clock period. SafetyScope assumes a default ratio of 1000 for state-element weightage unless overridden by the user. Since the customer chose to cover their state elements in their entirety in this design, *DC_transient* gets significant benefit.

The same safety mechanism applied in the same manner to various designs resulted in varying coverage numbers. Understanding the structural nature of the design allowed the user to re-target their safety mechanism accordingly to achieve their goals. Failure to do so would have merely set the user up for unpleasant surprises during the validation phase – whether via fault simulation or other techniques.

Coupling the ISO standard's baseline recommendation with a robust structural understanding (with the safety architect's functional view overlaid on top) is the only reliable way to meet your ASIL goal.

From Annexe C of ISO26262-11:2018

Austemper design and its automotive customers are not the only ones who have discerned the risks of a block-level analytical estimation of DC numbers. ISO26262's latest revision, the 2018 release has this to say about the computational accuracy of failure mode coverage and the granularity of analysis needed.

> … for a digital component at stand-alone level, a deeper analysis (e.g. at sub-part level) can be needed in order to compute with the required accuracy the failure rates and failure mode coverage of parts and sub-parts, to be used afterwards by system engineers. In other words, without an accurate and detailed digital component stand-alone level analysis, it can be very difficult to have good data for system-level analysis

Conclusion

Automotive electronics systems have dealt with functional safety using spreadsheet-level top-down safety analysis and component failure rates derived from past history and expert judgement. While deemed sufficient for bygone eras, the dawning age of self-driving automobiles and autonomous navigation impose a safety burden that can only be met with a ground-up analysis of the various elementary parts that comprise the system. The ISO-26262:2018 standard validates this via the all new part 11 that is devoted to semiconductor and IC-level analysis of functional safety. The hazards of sticking with the top-down approach are borne out by real-world case studies on currently shipping automotive IP and SoCs. In particular, different IP blocks react differently to the same safety mechanism based on a variety of characteristics. Performing a bottoms-up analysis is clearly outside the scope of traditional tools and calls for a structural-evaluation based tool. Austemper's ***SafetyScope*** is currently the only tool in the market that fulfills the requirement.

About Austemper

Austemper Design Systems provides the industry's only end-to-end tool suite to analyze, augment and verify functional safety in system-on-chip (SoC), application specific integrated circuit (ASIC) and intellectual property (IP) designs, ensuring they meet functional safety requirements. Austemper of Austin, Texas, was founded in 2015 by experienced semiconductor professionals with the mission to provide a best-in-class solution to meet functional safety requirements of the automotive, industrial, medical and enterprise markets. For more information, visit: www.austemperdesign.com