DVCON 2014 DESIGN AND VERIFICATION CONFERENCE AND EXHIBITION INDIA
Sept 25-26, 2014
Hotel Park Plaza
Bangalore, India

Compliance driven Integrated circuit development based on ISO26262

Haridas Vilakathara, Sr.System Architect, NXP Semiconductors, Bangalore, India
(haridas.vilakathara@nxp.com)

Manikandan panchapakesan, Architect, NXP Semiconductors, Bangalore, India
(manikandan.panchapakesa@nxp.com)

Abstract— ISO26262 is a standard addressing functional safety for road vehicles. One of the important features of ISO26262 is the safety concept based on fault reaction time or what is called as fault tolerant time interval. This paper describes a generic safety architecture proposal that can be applied at IC level to comply with the fault reaction time and diagnostic elements to meet the fault tolerant time interval requirements. ISO 26262 also emphasizes on the development process to take care of systematic faults or failures. This paper describes a requirement and compliance driven integrated circuit development methodology based on ISO26262 to create safe semiconductor integrated circuits for use in a road vehicle. The following aspects of safety development life cycle process are addressed 1) Concept of a safety life cycle on top of project life cycle. 2) Structure for a safety organization and safety life cycle management. 3) How to integrate a safety oriented life cycle on top a standard IC development life cycle based on "V-model" development cycle. 4) How to capture and mange requirements (DOORS based) in safety driven development process 5) Qualitative (FMEA) and quantitative (FMECA &FMEDA) risk based approach for capturing, tracking and analyzing safety requirements across the entire life cycle starting from requirement capture to IC validation.

A compliant driven development process implies that verification is performed throughout the entire IC/product development life cycle with clear ownership of activities by each specific party involved for each of the major work products such as requirement, design, integration etc. One hundred percent traceability to requirements and design units is another requirement from verification. Similarly validation also has a strict meaning by the standard to ensure the highest level of safety requirement and safety goals has been meeting and they are correct. This paper describes a safety extension to the verification and validation to address the above requirement. Apart from compliance driven verification, this paper also suggest the following as extension to improve the verification quality. These are constraint driven verification, design intent verification, risk based (inputs based on FMEA, FMECA) verification, fault injection testing. This paper also address on how to ensure the correctness of tools used for the design and verification process. This paper also emphasize the need for building a safety culture for sustainable development process and integrating the safety culture into the organization level methodologies and way of working. The paper concludes with a statement on the role of process assurance and documentation as key aspects in realizing a safe electronics for use in a road vehicle.

Keywords—component; formatting; style; styling; insert (Style: key words)

I. INTRODUCTION

ISO 26262 is a functional safety standard based on a more generic IEC 61508 standards for passenger vehicles. The standard provides a well-defined, automotive-specific safety life cycle, with functional safety in mind covering the product life cycle starting from concept phase after production phase. The standard follows a typical V-model for product development. The standard addresses system-level, hardware, and software sub-parts, each of which also follows the V-model. In this paper we describe a development methodology tailored to suit a semiconductor device development process.

ISO 26262 recognize two types of faults that can lead to failures within a system. This can be classified as one of two types:

• Random Faults

• Systematic Faults

Random Faults are due to physical causes and only apply to the hardware components within a system. This type of fault is caused by effects such as environmental conditions, operational parameter variations, and a possible misuse. It is possible to evaluate the failure rate and risk associated with such failures by gathering data through testing and previous history of failures.

Systematic Faults are produced by human error during system development and operation. This could be due to an error getting introduced into the design during specification, design and development stages. Similar to that an insufficient verification and validation techniques may result in an un-detected bug creeping into the product.

Table I. Integrated circuit failures and their causes

	Process gap analysis		
	Failure type	***Reason***	***How to prevent or reduce occurrence***
1	Random failure (life time failures)	Problems Inherent to semiconductor process Environmental conditions Disturbance in operations conditions Reasonable misuse and handling	Safety architecture for detection and management of random failure (manage fault)
2	Systematic failure	Bugs introduced during specification, design and development stages Un-detected bugs during verification and validation	Compliance driven process development and continues improvement (avoid fault) Use of best practices and process development methodologies

II. SAFETY ARCHITECTURE FOR MANAGEMENT OF RANDOM FAULTS

Given the fact that both random failures due exists in every integrated circuits, the ISO2626 recommend a mechanism to detect such failures during the operational life of the product through a set of diagnostic features for detection of such failures, and transfer the system to a fail-safe or fail-operational state based on the application severity with respect to safety. The following are the fundamental concepts implied in ISO26262.

1. It is impossible to develop a zero error (bug free) system due to
   a. Specification, implementation or realization errors
   b. Random hardware failure may occur due to reasonably foreseeable operational errors or reasonably foreseeable misuse.
2. It is possible to build a system with acceptable failure rate
   a. Acceptable failure rates vary per application
   b. Classified by automotive safety integrity levels (ASIL) and frequency and severity of functional failures
3. If a failure occurs, the system needs to be transferred into a safe state
   a. Failure event should not lead to unacceptable risk
   b. System must detect all faults that could lead to a hazardous event
   c. The fault reaction time to achieve safe state must be short enough to avoid any hazardous event

III. FUNCTIONAL SAFETY ARCHITECTURE

The functional safety architecture is primarily meant to address the random failures that can occur during the life time of the product. Since these kinds of failures are random in nature and can occur any time during the operational life e time of the product, the best way to address such failure is to add diagnostic features into the design to take care of the following.

1. Infrastructure to detect a fault
2. Infrastructure to diagnose and process the fault information and
3. A facility to take decision (solution) to address the detected fault.
   a. Repair fault
   b. Contain fault

Since a fault means non availability of IC function at least temporarily, all four actions described above need to be completed in a reasonable time so that the safety of the overall system is not compromised. This is called as fault tolerant time interval illustrated in figure [1], and the value depends on the application scenario.

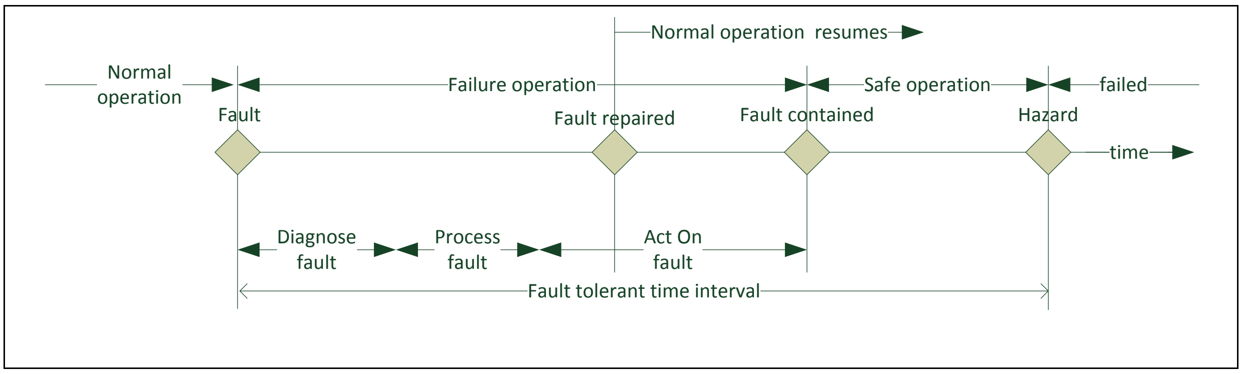


Figure 1. Diagnostic cycle

A. *Infrastructure to detect fault*

The infrastructure required for detection of a random hardware fault greatly depends upon the integrated circuit category and type. In a passenger vehicle category, this can vary from a

1. Front end electronic sensors with integrated analog & mixed signal circuitry along with a digital control circuitry
2. Microcontroller for engine, body or chassis control
3. Application specific integrated circuit for specific applications

The electronic sensor category mentioned many times contain high performance analog and mixed signal circuits such as ADC, DAC, and may even contain RF components. In such systems the basic fault detection circuits are built around a set of on chip sensors such as temperature monitors, voltage monitors, ADC clipping sensors, clock monitors etc. The microcontroller category is more prone to soft errors such as instruction memory corruption. In such cases additional circuits must be built to detect memory corruption, and resulting controller hung up situations. Memory error detection and correction logic, watch dog units etc. can add value here. In third category, typically contain multiple processing units along with local and shared memory for inter processor communications. Here apart from monitoring individual processor lock up condition, monitoring circuits must be built to monitor inter processor communication, and possible lock up conditions.

B. *Infrastructure to diagnose and process fault*

Once a fault condition is detected, a diagnostic procedure must be initiated and completed in reasonable time to assess the reason and severity of the fault. This requires some level of intelligence, and a most appropriate candidate for doing such action is none other than MCU control software. In the absence of a MCU as is the case with many sensor type integrated circuits, this can be done by explicit hardware diagnostic control FSM. However this can be quite expensive. In such scenario, the safety architecture for diagnosis of fault information can be built around the concept of using external MCU with full access to the fault detection units with the sensors.

C. *Infrastructure to take action against fault*

This totally depends on the safety integrity levels and safety class of the integrated circuits. The following are the few possibilities

1. Fail safe operation: This would be the minimum requirement from a functional safety view point. The functional behavior of the integrated circuit must be safe enough from a system view point. Typical example would be a simple electronic sensor, once a fault is detected, the sensor can be declared un reliable from a system view point, and the system can look for alternative ways to process the information. This may include sensor redundancy at system level.

2. Fail operational: In some of the highly safety critical scenario, a functional failure may not be acceptable. In this case the requirement would be to provide fully functional operational behavior against single point fault or functionality with reduced functions. This indirectly means redundancy at integrated circuit level.

IV. SAFETY ANALYSYS

Since life time failures of integrated circuits are random in nature, and in most cases the reason for failures can be found through life time test or qualification, it is possible to predict the probability of such failures through qualitative risk basement and quantify them through quantitative risk basement such as FMEDA (failure mode effect and diagnostic analysis). Combinations of both qualitative and quantitative assessments are recommended for accurate analysis.

A. *Qualitative risk assessment*

FMEA is considered as an efficient qualitative method in assessing the product level risk, starting at an early phase of the product development life cycle, and carried throughout the product life cycle and on a continuous evaluation basis. One of the critical aspects of the FMEA flow is the organizational level feedback loop, wherein the inputs are taken from organizational level quality history of similar products, and the FMEA findings are fed back to the quality history repository for future projects. The quality history consists of lessons learned in previous projects, field failure reports, benchmarking reports, and expert opinions etc. The FMEA flow and feedback mechanism is shown in figure[2]. The red lines on the left side of the picture indicate the feedback loop at the project level and the red lines at right indicate the organizational level feedback. FMEA validation is a key aspect at the organizational level; where in the effectiveness of FMEA is assessed through field failure reports of the product.

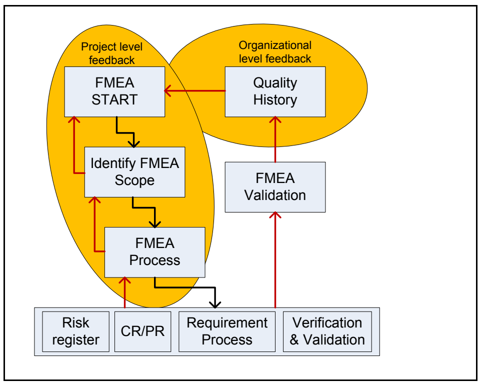


Figure 2. FMEA cycle

Within the project the FMEA is conducted at three levels. We will be discussing the first two FMEA items in this paper. The semiconductor manufacturing itself is considered as a matured process, and if not the Semiconductor fabrication house along with process flow validation team will be qualifying the process through various test chip programs.

Table II. FMEA at project level

	FMEA levels	
	FMEA scope	***process***
1	Concept level	Focuses on potential failure modes associated with the proposed functions or a concept proposal
2	Design level	Focuses on potential failure modes of products caused by design deficiencies
3	Semiconductor Process level	Focuses on potential failure modes of the semiconductor process that are caused by manufacturing or assembly process deficiencies

1) Concept phase FMEA

The primary strategy is to focus on ensuring right information at the conceptual design stage and then preserving the information integrity as we proceed through detailed design and implementation stage. We found that an early concept level FMEA is a good mechanism to critically evaluate the requirements itself and to validate the concept against the system constraints. Focus is on the interaction between systems at concept level and identifies potential failure modes caused by interactions. This can used to analyze concepts in the early stages before hardware is defined. The following are the benefits of doing an evaluation at an early stage.

1. Helps in selecting the optimum concept alternatives, or determine changes to design specifications. It also can identify system level testing requirements.
2. Helps in determining hardware redundancy and fault tolerance requirements based on failure modes and effects.
3. Helps to select the optimum concept alternatives, or determine changes to Design Specifications (DS).

The following are the outputs of the concept level FMEA that may influences the system level decisions

1. A list of potential concept level Failure modes and causes.
2. A list of actions (and to track) to eliminate the causes of Failure Modes, or reduce their rate of occurrence.
3. Decision on redundancy management (if required).
4. Specific operating parameters and boundaries as key specifications in the design phase.
5. New test methods or recommendations for new generic testing.
6. Decision on which concept to pursue.

2) Design phase FMEA

Here again apart from the standard design practices, clear attention is given on doing a design level FMEA based approach in identifying the weak spots in the design. The focus is on identifying potential failure modes of products caused by design deficiencies and the mission profile/boundary conditions of the design. The following are the generic guidelines followed in conducting the design level FMEA.

1. Analysis based hardware functions, interfaces, hardware-software interaction ect.
2. Consider environmental conditions and its impact on design (EMI/EMC, ESD, Single event upset etc.)
3. An identified failure mode may provide additional information to help plan thorough an efficient verification and validation programs.
4. It also establishes a priority system for design improvements, provides an open issue format for recommending and tracking risk reducing actions and future reference to aid in analyzing field concerns.

The following are the benefits of the design level FMEA that can also influences the design decisions

1. Aiding in the objective evaluation of design, including functional requirements and design alternatives.
2. Evaluating the initial design against non-functional requirements (example environmental conditions such as ESD, EMI/EMC etc.)
3. Providing additional information to aid in the planning of thorough and efficient design, development, and validation programs.
4. Developing a ranked list of potential Failure Modes according to their effect on the "customer," there by establishing a priority system for design improvements, development and validation and analysis.
5. Identify Implementation/verification priorities.
6. Providing an open issue format for recommending and tracking risk reducing actions by linking FMEA results CR/PR, risk register etc.
7. Providing future reference, e.g., lessons learned, to aid in analyzing field concerns, evaluating design changes and developing advanced designs.
8. Additional information to validate the system specification and V&V plan by linking FMEA items to V&V plan and to requirement management process.

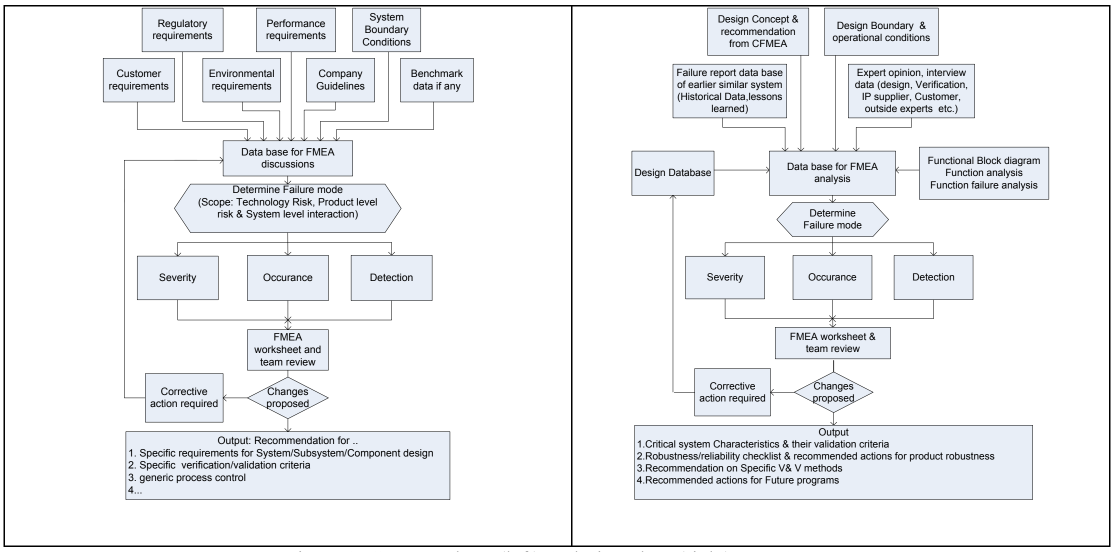
The following are the valuable outputs from a design FMEA process.

1. A list of potential product failure modes and causes.
2. A list of critical characteristics of the system to help in design priority setting
3. A list of recommended actions for reducing severity, eliminating the causes of product failure modes or reducing their rate of occurrence, or improving Detection.
4. Feedback of design changes to the design community

A critical issue with respect to FMEA is on how we start a FMEA process. The following can be considered as generic guidelines to get the right information on table to conduct an effective FMEA process.

1. Review specifications such as the statement of work (SOW) and the system requirement document (SRD), System configurations, designs, specifications, and operating procedures, Interface information and functional descriptions.
2. Analyze above with respect to key requirements
3. Compile information on earlier/similar designs from in-house/customer users such as data flow diagrams and reliability performance data from the company's failure reporting, analysis and corrective action system
4. Collect data by interviewing: architecture, design, verification, customer, IP suppliers and outside experts to gather as much information as possible

5. Create boundary condition diagram at system level (for Concept FMEA) and functional block diagram for Design FMEA
6. Identify the sensitive areas in integrated circuits. It is easy to start if the SRS/HRS specify safety requirements, If not start with a generic way, such as finding a sensible zone (a sensible zone is one of the elementary failure points of the IC in which one or more faults converge to lead a failure). Valid definitions of sensible zones are, HW–SW interface, memory elements, critical inputs and outputs, critical nets such as clock, complex IP/subsystems, and other key observation points


Regulatory requirements
Performance requirements
System Boundary Conditions
Customer requirements
Environmental requirements
Company Guidelines
Benchmark data if any
Data base for FMEA discussions
Determine Failure mode (Scope: Technology Risk, Product level risk & System level interaction)
Severity
Occurance
Detection
FMEA worksheet and team review
Corrective action required
Changes proposed
Output: Recommendation for ..
1. Specific requirements for System/Subsystem/Component design
2. Specific verification/validation criteria
3. generic process control
4...
Design Concept & recommendation from CFMEA
Design Boundary & operational conditions
Failure report data base of earlier similar system (Historical Data,lessons learned)
Expert opinion, interview data (design, Verification, IP supplier, Customer, outside experts etc.)
Design Database
Data base for FMEA analysis
Functional Block diagram
Function analysis
Function failure analysis
Determine Failure mode
Severity
Occurance
Detection
FMEA worksheet & team review
Corrective action required
Changes proposed
Output
1.Critical system Characteristics & their validation criteria
2.Robustness/reliability checklist & recommended actions for product robustness
3.Recommendation on Specific V& V methods
4.Recommended actions for Future programs

5. Figure 3. Concept phase (left) & design phase(right) FMEA

B. Quantitative risk assessment

Since majority of hardware failures are due to life time failure and random in nature, it is possible to predict the probability if failure occurrence and rate through statistical means. Also if there are means to find out fault during the operational life of the product, then it would be possible to reduce the overall failure rate at the system level. FMEDA (Failure mode effect and diagnostic analysis) is one such technique that can be used at IC level to compute the random failure metrics such as single point fault metric (SPFM) and latent fault metrics (LFM). In principle the FMEDA can be exercised at product level based on the following information.

1. Based on Diagnostics coverage
   a. Ratio of detectable failures probability against all failure probability
   b. Diagnostic or self-checking elements modelled
      i. Complete Failure Mode Coverage
      ii. All failure modes of all components must be in the model
2. Failure Mode Classifications in FMEDA
   a. Safe or Dangerous: Failure modes are classified as Safe or Dangerous
   b. Detectable : Failure modes are given the attribute DETECTABLE or UNDETECTABLE
   c. Four attributes to Failure Modes: Safe Detected(SD), Safe Undetected(SU), Dangerous Detected(DD), Dangerous Undetected(DU)
3. Goal : Statistical Safety: based on Safety Integrity Level (ASIL)

In addition, the FMEDA need to be supplemented with information from FMEA to associate fault sensitivity information based on function criticality. This would provide reasonable scaling facture while computing individual component failure data. Similarly field failure data from previous product of life time test data during product qualification can be used for computing the FMEDA metrics more accurate.

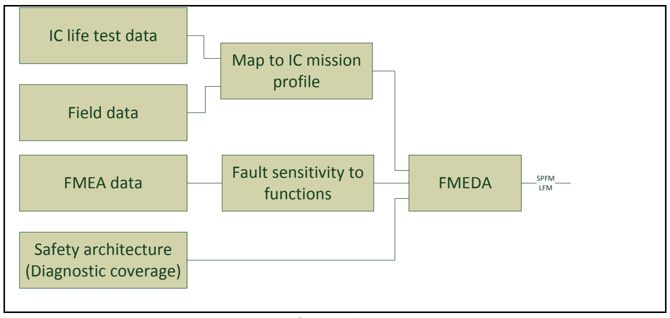

IC life test data
Map to IC mission profile
Field data
FMEA data
Fault sensitivity to functions
FMEDA
SPFM
LFM
Safety architecture (Diagnostic coverage)

Figure 4. FMEDA process

V. DEVELOPMENT PROCESS FOR THE MANAGEMENT OF SYSTEMATIC FAULTS

Traditionally it was assumed that majority of hardware failures are random in nature and software failure are systematic in nature and there is no reason to consider systematic failure types in a hardware only context. However, this is changing with the growing complexity of integrated circuits. Today high level languages are used in describing complex integrated circuit. This means there is a possibility to introduce an error or bug into the design while specifying the system at a higher abstraction level or while translating the high level specification to the actual hardware through various design translation process.

Since it is not possible to statistically predict the probability of systematic faults, it is not possible to quantify the associated risks. Hence the arguments for acceptability of using a complex hardware system are based on the suitability and maturity of a development life cycle model.

The key item of the life cycle is the central planning and control function along with a concurrent process support functions based on the following aspects.

1. An organizational level product creation process defining clear milestones and gating criteria within the project execution life cycle.
2. Requirement driven development flow, wherein requirement management and traceability across work product is considered as key to project success.
3. A concurrent supporting process to ensure process and product assurance along with 100% forward and backward traceability to the requirement process.

An effective planning process can define a number of gates/milestones, which divide the project up into manageable project phases. Within these project phases activities will be planned to generate a number of deliveries. Each of these deliveries will have their own maturity. It is related to the type of delivery, i.e document, design, hardware, software, etc. based on the maturity model is applied. The longer implementation phase is further sub divided into different phases based on product maturity expectations. At the end of each planned phases, formal reviews and audits are conducted to make sure that the expected process compliance and product level maturity are in place.

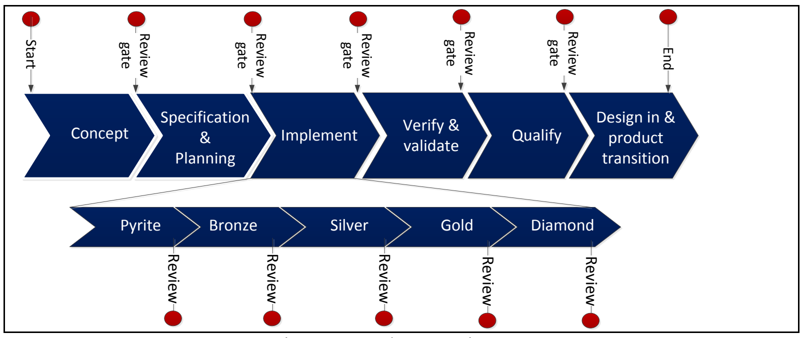


Figure 5. Product creation process

Table III. Implementation sub phases

	Implementation sub phases	
	Phase	***Key attributes***
1	Pyrite	Identify the key IP's, components and their sources. Risk assessment
2	Bronze	Early prototype state. Freeze the top level and component level interfaces. All individual IP's are Silver quality. Interface control for any further module level interface changes
3	Silver	Freeze all IP's. Basic functional verification OK. All IP's are of gold quality
	Gold	RTL freeze. Functional coverage 100 %

A. Compliance driven development process with safety extensions

An ISO26262 compliant development flow recommend a 'V' model project life cycle along with a dedicated safety management function. From an implementation perspective, this can be translated into the following

1. Safety life cycle on top of standard "V" model with clear input, output and dependency to work products within the project
2. A dedicated safety organization function at organizational level as well as at the project level to keep track of safety related activities

B. Safety life cycle on top of "V" model

The key objective is to link the safety related activities into the standard project plan with clear input, output and dependency to the work products within project. This is shown in figure[7].

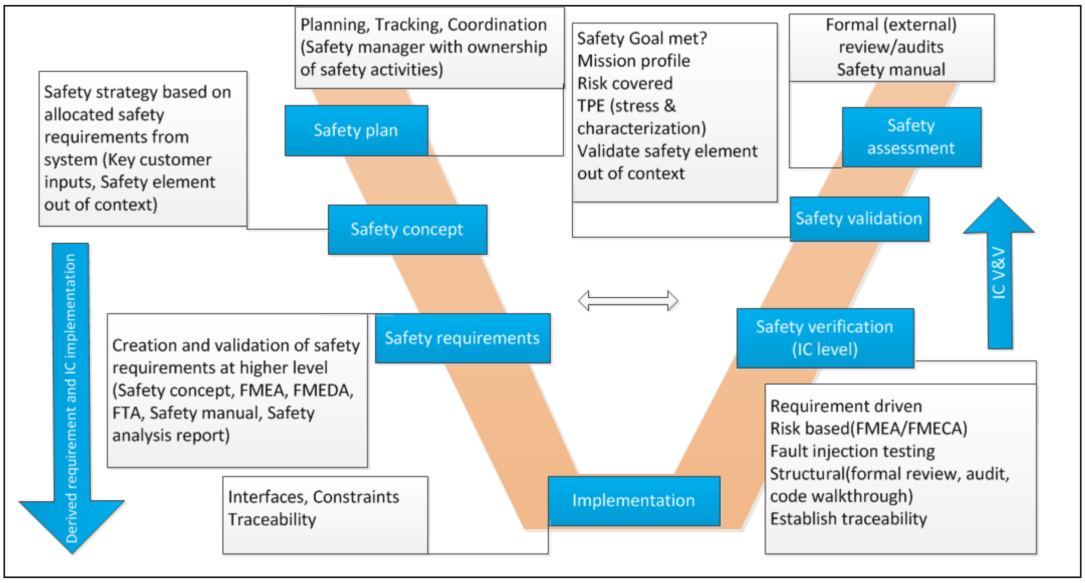

Planning, Tracking, Coordination (Safety manager with ownership of safety activities)
Safety Goal met? Mission profile Risk covered TPE (stress & characterization) Validate safety element out of context
Formal (external) review/audits Safety manual
Safety strategy based on allocated safety requirements from system (Key customer inputs, Safety element out of context)
Safety plan
Safety assessment
Safety concept
Safety validation
IC V&V
Derived requirement and IC implementation
Safety verification (IC level)
Creation and validation of safety requirements at higher level (Safety concept, FMEA, FMEDA, FTA, Safety manual, Safety analysis report)
Safety requirements
Requirement driven Risk based(FMEA/FMECA) Fault injection testing Structural(formal review, audit, code walkthrough) Establish traceability
Interfaces, Constraints Traceability
Implementation

Figure 6. Safety life cycle on top of project life cycle

Table[IV] describes the major extension of activities to the standard 'V model development process.

Table IV. Safety extension to project life cycle

	Safety extension to project		
	Standard project	***Safety extension to project***	***Proof of safety***
	Project plan		
1	Project planning Project organization Development process	Safety life cycle activities (Safety org, Safety manager, safety plan, tracking & assessment) Safety team organization & management	Safety case and Safety arguments
	Requirement management		
2	Requirement capture & requirement engineering, FMEA(qualitative)	Derive Safety goals, safety requirements, safety concept, and safety requirements and methods. FMEA, FMEDA & FTA	Safety goals, Safety concept Safety requirements with proof of traceability FMECA/FMEDA/FTA reports Safety manual
	Verification & validation		
3	Product assessments V&V Reviews/audit	Safety V&V, Safety assessment, Tool qualification	Safety V&V reports Safety assessment reports Validation reports (proven in use argument)
	Configuration management and change control		
4	Work products Baseline management Change management	Safety case as CI Impact analysis on safety functions(for CR)	Baseline for safety case & safety case reports Traceability and impact analysis report liked to requirement management
	QM & process assurance		
5	Quality manual Template/guidelines, Checklist/examples	Safety culture, Project level tailoring, Continuous improvements Reviewing and tracking compliance with plans. Identifying and documenting deviations from plans.	Tool confidence report, Evidence on safety culture deployment, Project tailoring report. Qualification reports, Risk assessment reports, Periodic confirmation review reports

C. *Safety organization*

The primary objective of having a separate safety organization function within the project execution is to collect evidence to prepare safety case arguments to support the following

1. Product claim (random faults): The key metrics such as SPFM, LFM, etc. against the safety class requirement
2. Process claim (systematic faults): To provide confidence in the process development flow along with process compliance metrics

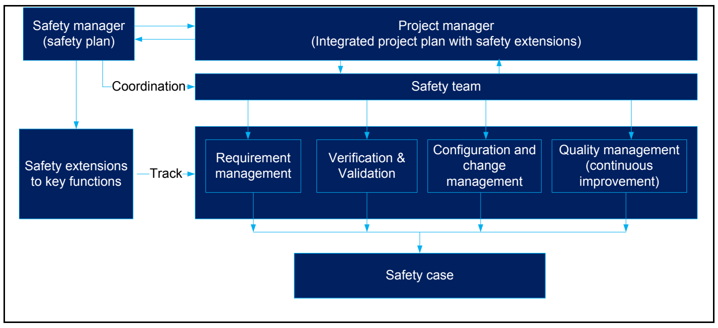

Safety manager (safety plan)
Project manager (Integrated project plan with safety extensions)
Coordination
Safety team
Safety extensions to key functions
Track
Requirement management
Verification & Validation
Configuration and change management
Quality management (continuous improvement)
Safety case

Figure 7. Safety organization

One of the important observation we can make w.r.t the process claim is that, the errors introduced into the work product during the project execution are not random in nature. Hence cannot be supported with any statistical data. It is more of building confidence into the process flow through an organizational culture with functional safety in mind. It is very difficult to implement a safety critical product if the organization is not prepared for it. A well-defined quality management function that focusses on continuous improvement within organization and at the project level is the key to the success.

Within a project, the process of building a safety critical product can be categorized into the four major area.

1. Requirement management
2. Design integration
3. Verification and validation
4. Configuration management and change control
5. Quality management and process assurance with continuous improvement

1) Requirement management

There are two important aspects in a requirement driven product development flow.

1. There must be a formal agreement with the system development process to asses and evaluate the underlying hardware requirements, and a mechanism to validate them
2. There must be an effective mechanism to track the requirement throughout the product development phase to various design and verification items.
   a. All requirements can be tracked to a design, verification and validation item
   b. No design element in repository without an assigned requirement
   c. Any PR raised can be tracked to a corresponding requirement ID
   d. Impact on existing requirements can be identified for CR

Automation and tool support is crucial to avert any trivial human errors introduced through manual control and management of requirements. In this project we used commercial requirement management software from Telelogic named "DOORS". The following are the critical aspects of requirement management that can be easily managed through such tool based requirement management.

Table V. Requirement attributes

	Requirement attributes	
	Attribute	***Description***
1	Valid	Can map to either a customer requirement or to a organizational level guidelines
2	Traceable	Lower level requirements are derived from high-level ones, and to a design

	Requirement attributes	
	Attribute	*Description*
		verification item
3	Complete	No user requirements are omitted
4	Consistent	No conflicting requirements
5	Relevant	No inefficient use of resources
6	Unambiguous	Less likely to lead to misunderstanding
7	Verifiable	All requirements can be mapped to a verification item

The most important aspect of organizing the requirement with valid attributes is to establish clear requirement traceability across work products. To accomplish this DOORS bases requirement management system need to be linked with the design data base. This is illustrated in figure [8].

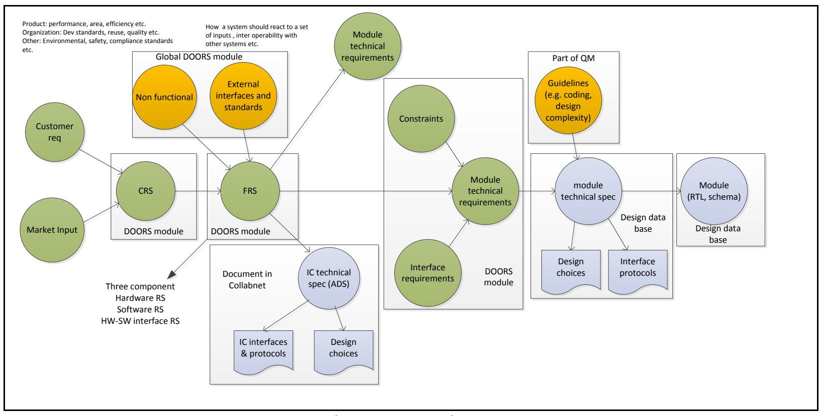

Product: performance, area, efficiency etc.
Organization: Dev standards, reuse, quality etc.
Other: Environmental, safety, compliance standards etc.
How a system should react to a set of inputs , inter operability with other systems etc.
Global DOORS module
Non functional
External interfaces and standards
Module technical requirements
Customer req
Market Input
CRS
DOORS module
FRS
DOORS module
Three component
Hardware RS
Software RS
HW-SW interface RS
Document in Collabnet
IC technical spec (ADS)
IC interfaces & protocols
Design choices
Constraints
Module technical requirements
Interface requirements
DOORS module
Part of QM
Guidelines (e.g. coding, design complexity)
module technical spec
Design data base
Design choices
Interface protocols
Module (RTL, schema)
Design data base

Figure 8. Requirement tree

By organizing the requirements through a formal process we are making sure that, we have only one source for to capture and mange requirements, thereby enabling easy traceability across work product. Further to this the following aspects of product quality assurance can be easily established through the following.

1. Requirement traceability across work products (Forward and backward traceability between requirement – design – verification).
2. Formal review and audit process across work products, making sure that the requirements are correctly mapped to at least one design elements.
3. System level requirements can be allocated to sub modules/IP, and they can be tracked through separate IP projects or can be realized through a proven re-usable IP component.
4. Formal alignment with system/SW requirement process.

A typical integrated circuit system may contain many IP components that get integrated at different hierarchy levels. All these components are to be typically customized (configuration) to meet the architectural requirements. Configurable IP offers a solution to the problem above by allowing the system integrator to set up configuration parameters through a script that configures the block according to the parameters. However to implement such feature in an automated way, the basic configurable architectural intellectual property (IP) blocks are needed in a standardized (Standardized views are required in IP deliveries, documents, and an electronic description of the IP etc. such as IP-XACT view) machine-readable form, so that this can be pushed into an automated design and verification flow. The system integrator can evaluate the IP configurations against the system requirements by quickly integrating the system and evaluating the same. If the design requirements are not met, then different configurations of the IP can be tried. This provides options to the integrator in analyzing the requirements and how it matches with the IP configuration parameters.

Automation in standard design environment need to be considered in realizing the project that addresses the requirements listed above through extensive utilization of architecture, IP reuse, efficient system integration, hardware-software (HW-SW) co-verification and design flow automation. This enables early RTL integration, thereby allowing evaluation against hardware requirement at an early development phase. Automation also allows us to have quick early iterations of the hardware, and if the iterations are planned properly, this will enable in achieving minimizing overall design time with sufficient product maturity. The following are the few important aspects of design iteration loops

1. Quick early hardware for early trials (bronze phase) to analyze the impact on crucial hardware parameters such as chip area, DFT, power, performance etc.
2. Strict formal releases towards Silver/Gold
3. Promote local iterations in activities such as Coding, verification, synthesis timing closure etc. at smaller blocks (divide and conquer)
4. Promote IP reuse, where ever applicable

One of the important parameter that need to be noted for successful integration is that , we need to make sure that the integrated IP is of right quality and maturity and this can be done through a formal review and audit process called as IP incoming inspection.

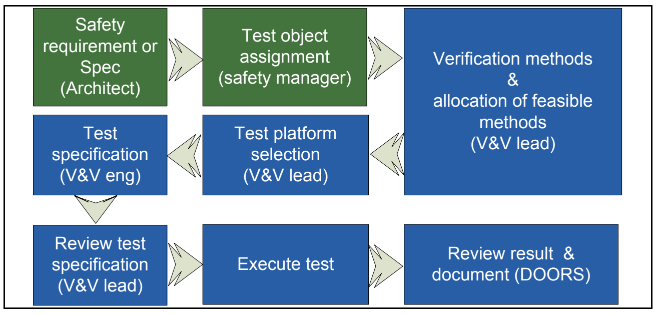
1. Do an IP vendor and IP repository assessment
2. Expert team (Architects, SW experts) review on IP configuration, IP Reuse status, and IP maturity status (version, CR/PR database, silicon proven status etc.), standardized views (IP_XACT), interconnect standards, naming, signal convention, clock and reset, global control registers, Hardware-Software interface etc.
3. IP documentation, specially from an IP-SoC integration perspective

2) Verification & validation

a) Verification process

In a requirement driven SoC development approach, the hardware design and verification activities need to be done independently. The hardware designer works to ensure the design of the hardware will meet the stated requirements. Similarly, the verification engineer will generate a verification plan which will allow for testing the hardware to verify that it meets all of its derived requirements. The verification process provides assurance that the hardware item implementation meets all of the hardware requirements, including derived requirements. Apart from requirement coverage the most important goal of verification is to ensure the consistency of requirement specification artefacts across the design translation process. In summary we can state the following as the major goals of verification

1. Checks consistency of the requirements specification artefacts and other development products
    a. Req->design->RTL/schematics-> netlist-> Back End ->GDS2
2. Safety not compromised during design translation process
    a. Through verification flow involving safety team (fig) with clear ownership on who does what
3. Each requirements can be associated with at least one verification item
4. Establish traceability to the requirements
    a. Any design item can be tracked to a requirement ID
    b. No design element exists in design data base without having a parent requirement ID


Safety requirement or Spec (Architect)
Test object assignment (safety manager)
Verification methods & allocation of feasible methods (V&V lead)
Test platform selection (V&V lead)
Test specification (V&V eng)
Review test specification (V&V lead)
Execute test
Review result & document (DOORS)
Figure 9. Safety oriented verification flow

5. The following are the scope for a compliant driven verification
    a. Coverage: Link coverage item to requirements with evidence(100% requirements coverage) In addition to other standard coverage metrics
    b. Test plan, test item, test results linked to DOORS to establish traceability
    c. Verification at every stage of data translation (req-design-RTL-netlist-post layout)
    d. Constraints driven
        i. Interface constraints(limit, boundary value etc.), rules, design constraints such as Boundary value analysis, negative testing, control flow (protocols), data flow (protocol), overflow, underflow
    e. Design intent verification
        i. Intended operational environment and rules
        ii. Temporal behavior to stimuli (protocol sequence, error recovery
    f. Risk oriented
        i. Inputs from FMEA/FMECA
    g. Fault injection testing
        i. Coverage for safety architecture components & diagnostic capabilities
    h. Structural
        i. formal review, audit, code walkthrough

b) Validation process

The validation process provides assurance that the hardware items primary and derived requirements are correct and complete with respect to system requirements allocated to the hardware item.

1. Requirements Validation (right product) through proven in use arguments (use case validation under realistic environment around)
2. Check IC requirements specification against customers goals and requirements
3. Provides assurance that the hardware item derived requirements are correct and complete with respect to system requirements allocated to the hardware item
4. Ensure that the highest level safety requirement, the safety goal, has been met and that they are correct
    a. The product is safe to use within the mission profile
5. Validation scope
    a. Check whether safety goal is met
        i. Customer driven or based on safety element out of context
        ii. Confirmation check on safety goals, safety architecture against functional safety of the items at intended use case level
    b. Use case validation
        i. Question/review safety goals from a use case perspective
        ii. Compliance with governing standards and other regulatory requirements
            1. Interface standards, EMI/EMC etc.
    c. Fault injection testing
        i. Validate safety architecture and diagnostic features
    d. Evidences against all the above

3) Tools and infrastructure

Tool assessment is an important step to make sure that hardware design and verification perform correctly. Suitability of the tool for their intended tasks are specified so that the process is traceable and repeatable. He following is the primary driving factors towards tool qualification

1. Why tool qualification
    a. IC design involves many translation process, and tool in general has the capacity to introduce error during various translation process
        i. e.g. RTL-> Synth netlist -> post layout
    b. Verification tools may fail to detect errors in the hardware items
2. Tool qualification makes sure that tool correctly functions (improve confidence in tool function)
3. The library components and different views used during the design translation process also need to be of mature quality and qualified one

The following (figure[10]) can be used as a methodology or flow for tool qualification

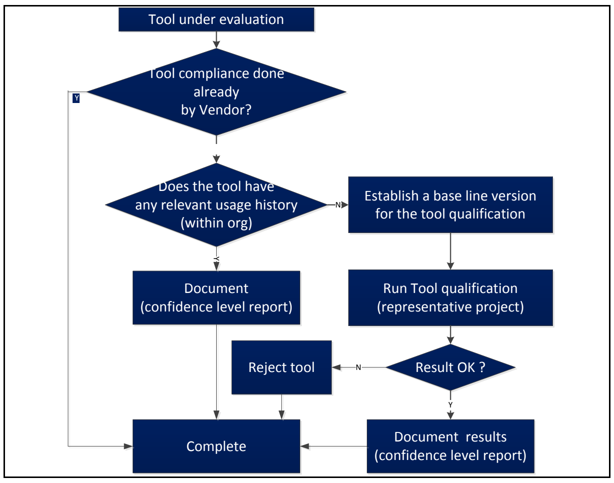

Tool under evaluation
Tool compliance done already by Vendor?
Y
Does the tool have any relevant usage history (within org)
N
Establish a base line version for the tool qualification
Y
Document (confidence level report)
Run Tool qualification (representative project)
Reject tool
N
Result OK ?
Y
Complete
Document results (confidence level report)

Figure 10. Tool qualification flow

Apart from tool qualification IC development infrastructure also can plays a bigger role in maintaining data integrity and maintainability. An early assessment of the tools used along with periodic planned review and audit is necessary to make sure about the data base integrity and correct transformation of the requirement to the product transition stage. The following are the typical elements that need to be reviewed and assessed within a project.

- Focus on effort spend up front to prevent design problems due to design environment
- Infrastructure review Scope
  - Data base structure
  - Development standards & flows
  - CM, CR/PR system and tracking methodology
  - Effective simulation analysis and regression systems
    - e.g. Automation in simulation/synthesis log analysis.
  - Communication between architects, design engineers, & verification engineers
    - Documentation templates, availability & traceability
- Planned audit
  - Collect the documentation maintained w.r.t database
    - Check all the team members in the team get right information at right time
    - Establish inks & traceability
  - Audit on CR /PR tracking
    - Defect density
    - CR/PR flow
  - CM audit

4) Configuration management & change control

One of the important aspects of configuration management is the management of design and verification data throughout the life of the product including design in and customer support. Hence, there is a need for an effective recording and configuration management system. There are several interrelated aspects to configuration management. The primary requirement is a consistent repository of data. This repository contains identification of the design environment, a collection of design artifacts sufficient for consistent replication, and verification data that provides sufficient evidence that the design meets its requirements. Information in this repository needs to be maintained such that controlled changes preserve a consistent set of data. The following are the key points that need to be considered with respect to configuration management.

1. Configuration management plan aligned with ISO TS 16949, ISO9001
   a. Maintenance of safety case as a living argument to provide a mechanism to establish the interdependency between the elements of safely case
2. Change management
   a. Establish link between change management & problem reporting process and requirement management (through DOORS)

3. Safety representative in CCB
    a. Ascertain the impact on functional safety w.r.t change requests

5) Quality management and design assurance

Design assurance is an integral part of all the activities mentioned above. At a formal level the design team along with the quality assurance officer will analyze the following critical aspects at appropriate gate/milestone

- Formal reviews & audits
  - By external experts
    - IP Vendor & IP quality checklist (IP reuse & IP intake strategy)
    - Technology Library
    - Process Reliability measures
  - Project level
    - Data base (CM, CR/PR, Documentation)
    - Compliance check against FRS/Impl/Verification
    - Verification strategy/methodology review
    - Verification coverage review
    - Explicit SoC constraints review
    - Non functional requirement review (coverage)
    - Layout review by a separate team
    - Pin/PAD list

The following need to be considered as a safety extension to the quality management function

1. Process Gap analysis (safety)
2. Safety culture within organisation & project team
3. Process tailoring with respect to the project
4. Process assurance
    a. Design data base management
    b. Design assurance

a) Process gap analysis

The primary objective of gap analysis is to assess the safety culture within the organization with respect to product creation and project execution by looking into the key process functions such as requirement management, configuration and change control management, quality and process assurance. The assessment need to be done prior to the execution of a project, and continually monitored through an audit process. The flow to be used for such gap analysis is depicted in figure ….

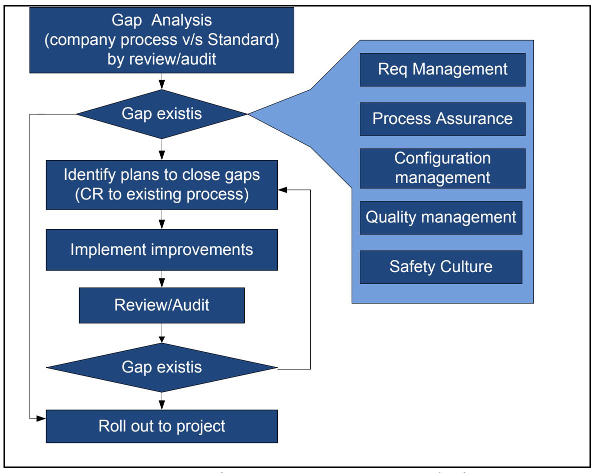


Figure 11. Process gap analysis

The key functions that need to be looked for during a gap analysis review are given in table [VI]

Table VI. Process gap analysis

	Process gap analysis	
	Item	***Gap to look for***
1	Requirement engineering	Demonstrate process for requirement capture and to establish traceability to design and verification
2	Process assurance	Reviewing and tracking compliance with plans. Identifying and documenting deviations from plans. Determining if transition criteria have been met between lifecycle stages.
3	Configuration & change management	Demonstrate supporting processes with respect to safety for Change Management & its impact on requirements, design or verification
4	Quality management	Quality manual update for safety life cycle process, methodologies, Supporting process (e.g. check list, templates guide lines for FMEA/FMEDA)
5	Safety culture (part of quality management)	Evidence of management of safety life cycle activities at organizational level and at project level. Organization chart showing safety role, Competence management (training & qualification program) & safety assessment

b) Safety culture

ISO26262 considers safety culture/climate assessment at organizational level. It defines the new development process requirements for a safety oriented work culture. There are four areas to be considers 1) The product creation/business decision environment 2) the project work culture to drive requirement oriented product development (as defined by the “V” model), 3) the knowledge gap on how to shift from qualitative to quantitative product reliability assessment and 4) the time and awareness to manage the implementation of the additional safety confirmation measures. Table [VII] few such scenario’s as a comparison between standard development process and safety oriented process at a project level.

Table VII. Safety culture

	Safety culture	
	Normal process	***Safety oriented process***
1	Safety measures are not planned. No Specific budget for safety	Necessary measures are planned according to safety plan (planning, tracking, coordination). Safety plan is integrated into project plan (receivables, deliverables, and resources)
2	Requirement capture and management are based on expert opinion (key architects drive them)	Structured requirement management with bi-directional traceability. Safety requirements are identified separately with associated ASIL level
3	Risk analysis & tracking is done, but may not structure (updated on regular basis)	Structured and quantitative risk analysis is a must at beginning of project, and is continuously updated with evidence
4	Typically only a FMEA cycle is followed (qualitative analysis)	Both qualitative (FMEA) and quantitative (FMECA/FMEDA,FTA) analysis is a followed at beginning of a project and is continuously updated with evidence
5	Change request are accepted at any stage (primarily look only into schedule impact)	Change requests are analyzed and evaluated against functional safety and accepted only through a strict formal change management process (four eye principle) and impact analysis
6	Safety assessment not a must	Safety assessment is a must and evidence to be preserved until end of life

I. CONCLUSION

The key argument from this paper is that realizing functional safety in complex integrated circuit requires thinking from two angles.

1. A safety architecture with sufficient diagnostic capabilities to take care of life time faults inherent in hardware.
2. A well-defined development process with continuous improvement in mind to take care of systematic fault.

We presented generic safety architecture with FMEA and FMEDA as key techniques in analyzing the safety architecture and how to implement an effective FMEA and FMEDA within IC development project.

We described a compliance driven, development methodology that can effectively track the requirements, assess the risk at appropriate level and ensure process and product quality assurance across the product life cycle. Right information at right time at all level is the key for first time success

[1] ISO26262, International standard road vehicles - functional safety, 2011

[2] RTCA, 2000, DO-254: Design Assurance guidance for Airborne Electronic Hardware, RTCA, Inc., Washington, DC