2014 DESIGN AND VERIFICATION DVCON CONFERENCE AND EXHIBITION EUROPE

Requirements driven Verification methodology (for standards compliance)

Serrie-Justine Chapman
Test and Verification Solutions TVS
Engine Shed
Temple Meads, Bristol
BS1 6QH
T: +44 (0)7854 785 747
serrie@testandverification.com

Darren Galpin
Infineon Technologies UK
Infineon House, Great Western Court,
Stoke Gifford,
BS348HP
T: +44 (0)117 952 8754
Darren.Galpin@infineon.com

Dr Mike Bartley
Test and Verification Solutions TVS
Engine Shed
Temple Meads, Bristol
BS1 6QH
T: +44 (0)7854 785 747
mike@testandverification.com

ABSTRACT

Requirements-driven verification is based on ensuring that feature-level requirements are adequately verified by tracing such requirements through to verification tasks. It is similar to Coverage-driven Verification in the sense that it is metric-driven but differs significantly because the metrics derive from requirements rather than verification goals. Requirements-driven verification is also required for compliance with the increasing number of standards that control development of hardware for domains such as automotive (ISO26262) and avionics (DO254).

Keywords

Management, Measurement, Documentation, Design, Reliability, Standardization, Languages, Verification.

1. INTRODUCTION

In this paper we give an overview of a Requirements Driven Verification and Test (RDVT) methodology and explain how this methodology can be used to support compliance to various hardware (and software) development standards. We also demonstrate how advanced verification techniques can be deployed in RDVT. We describe an automated solution using an SQL database approach to capture the requirements tree and mapping to the metrics typically generated by advanced verification techniques.

2. Requirements Driven Verification and Test (RDVT) Methodology

The Requirements Driven Verification and Test (RDVT) methodology enables project progress to be analyzed and managed by accumulating data on the status of verification and test metrics over the duration of the project and automatically relating these back to the specified requirements. In this way every functional requirement can be mapped to a proof of implementation. Additionally any verification and test activity not relating to a requirement can be identified and questioned.

3. Development Standards Compliance

There are a number of standards that mandate various aspects of hardware (and software) to be used in a variety of industries. For example:

- IEC61508: Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems
- DO254: Design Assurance Guidelines for Airborne Electronic Hardware
- ISO26262: Road vehicles – Functional safety

A key feature of safety standards is that they aren't prescriptive, they don't mandate how you develop the hardware (and software), which lifecycle to use etc. Instead they set a number of process objectives and outputs for the generic development processes of planning, requirements, design, code, configuration management, assurance and of course, for verification

These objectives vary according to the required integrity level or class of the software/hardware. This is the second aspect of the standards. When the hardware and software is assessed for whether it can cause a hazardous system state it is assigned an integrity level which reflects the severity of the hazard. For example, ISO26262 has four ASIL (Automotive Safety Integrity Level - ISO 26262-1:2011 Road vehicles -- Functional safety [7]) A to D, with D indicating that the consequence can be catastrophic; ASIL A indicating a controllable, uncommon and relatively harmless hazard. It also has a QM (Quality Management) level which can be assigned to any features that are deemed to be non-safety related.

A hazard analysis of the system will determine the safety integrity level and that level will in turn reflect the objectives that must be met and the outputs that must be produced.

The key elements that all the safety standards commonly require to be produced are as follows:

- Plans and standards including requirements, design and coding standards, development plans. Change and configuration management, verification plans etc.
- Requirements should be specified, documented, validated, verified and managed. These may be specified with differing formats dependent on their Integrity level (formal, semi-formal, quality criteria etc.).
- Design specifications should be produced with the details varying by safety integrity level Specification of Safety requirements in all domains, technical, software, hardware, system.
- There should also be proof that reviews and analyses of outputs have been undertaken, recorded, available for audit and under configuration management.

- Well-argued proof of correct implementation of requirement specifications, to have been carried out to the appropriate level within the product, this may be through test, verification, documentation, proven in use or manual proof (i.e. review).
- Specific test coverage criteria are required to be met (e.g. line coverage. MC/DC coverage)
- Requirements should be traceable down through the design to the tests.
- Finally, for the higher integrity levels there must be independence in verification activities – the person producing an item cannot review or sign it off.

This paper will concentrate on the requirements management. For Requirements Management, ISO 26262 Stipulates

> *"The management of safety requirements includes managing requirements, obtaining agreement on the requirements, obtaining commitments from those implementing the requirements, and maintaining traceability."*

This leads to the following type of activities

- Requirements engineering, looking after requirements at a hierarchical level, which must be of good quality.
- Requirements mapping which also ensures there is no loss, incorrect translation or loss of context throughout the Requirements tree.
- Also the requirements need to be proven to be implemented and working

DO254 identifies a number of data items that must be produced including hardware traceability data defined as:

> "*Hardware traceability establishes a correlation between the requirements, detailed design, implementation and verification data to support configuration control, modification and verification of the hardware item*"

The above ISO26262 and DO254 stipulations traditionally lead to the type of requirements shown in Figure 1 "Typical Requirements Tree". However, with advanced verification techniques the mapping to "Proof of Implementation" is not always so straight forward.

4. Supporting RDVT with Advanced Verification Techniques

Hardware verification engineers tend to use a range of advanced verification techniques such as:

- Constrained random verification with automated checks based on models or scoreboards, etc, .
- Coverage driven verification based on functional coverage models and code coverage metrics, .
- Assertion-based verification.
- Formal property based verification and manual sign-off, review, directed tests etc.

A requirement might be signed off at multiple levels of hierarchy during the hardware development. For example, consider a design with a number of serial interfaces including SPI. We might want have requirements to cover the accesses allowed by the interface, specific hardware requirements, requirements relating to software accesses and a requirement that an interrupt is generated within 2 milliseconds of data arriving in a SPI.

At block level:

- The verification of the SPI block might involve a large number of constrained random tests.
- Functional coverage would ensure that suitable tests had been performed including, for example, 7 and 10 bit reads and write on the interface; specific hardware requirements such as clocking schemes, slave select, etc.;
- Assertions would help to ensure the correct behavior and assertion coverage would help to ensure they were adequately exercised.

At subsystem level:

- Formal verification connectivity checks would be used to ensure that the SPI is correctly integrated at subsystem level (including correct connection of the interrupt signal)

At system level:

- Directed tests running on the host CPU access registers in verification environment and back-door accesses using UVM-RAL provide a checker for those accesses.
- Those directed tests send data into the SPI and then software running on the embedded processor detects arrival of the interrupt within the allowed number of clock cycles.

An example partial hierarchy for these is captured in Figure 4, section 10 "Appendix A: Sample Hierarchy".

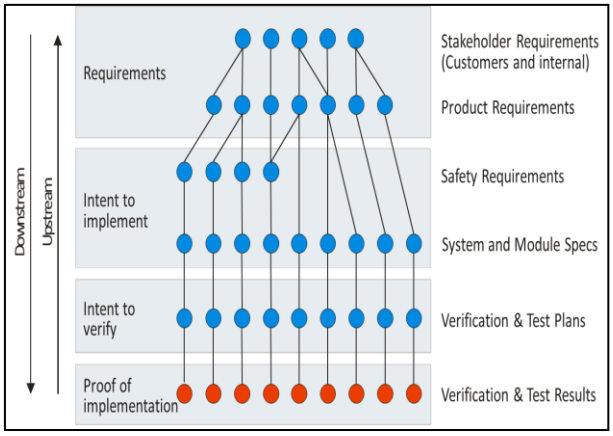

Figure 1: Typical Requirements Tree

Thus we can see that the one-one mapping implied by Figure 1 "Typical Requirements Tree" is unlikely to work with advanced verification techniques and hierarchical verification. Instead, the mapping shown in Figure 2 "Requirements Tree for Advanced Verification Techniques" is more likely.

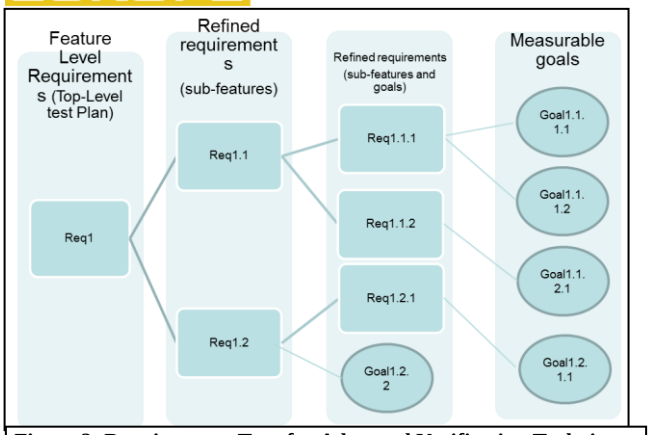

Figure 2: Requirements Tree for Advanced Verification Techniques

This leads to a different way to capture the requirements mapping and functional verification results which is discussed in the next section.

5. A new solution for RDVT

TVS has been working on a new solution for supporting RDVT for Infineon within the CRYSTAL project using advanced, hierarchical verification techniques. At the centre of this solution is an SQL database as shown in Figure 3 "Requirements Signoff Flow for Advanced Verification".

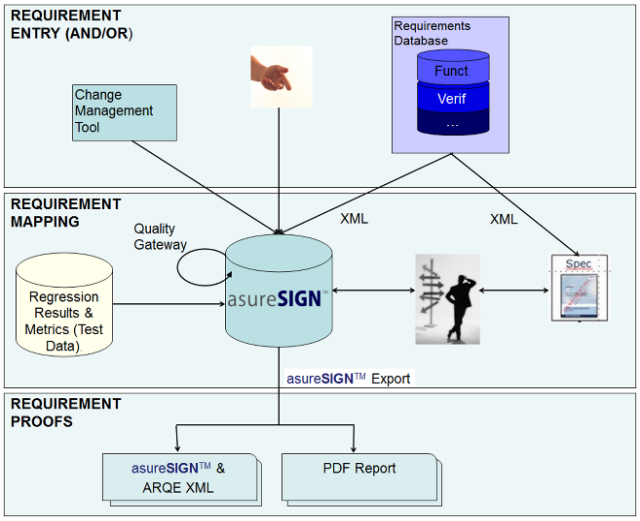

Figure 3: Requirements Signoff Flow for Advanced Verification

The database is able to capture the requirements tree shown in Figure 2 "Requirements Tree for Advanced Verification Techniques".

5.1 Database Design

In this section we give guidance on the type of data that needs to be stored in the database (shown as asureSIGN™ in Figure 3 "Requirements Tree for Advanced Verification Techniques"). We deploy RDBMS (Relational Data Base Management System) technology. Below are some examples of the key data required.

- Captures all the requirements from top level down to atomic level including the hierarchical relationship between them.
- Captures the bidirectional hierarchical relationship between requirements, features and verification goals.
- Capture all of the coverage and test information from different regressions, and the associated meta-data.
- Source from where the coverage are recorded with timestamp, user, version control system (like Git, ClearCase, SVN)
- Errors and error messages associated to test executions.
- Operation or request from user, and what was the outcome. This is used to provide audit trails.
- Captures user defined interim milestones (based on subset of full coverage goals) to measure progress of project.

5.2 Overview of Requirements Signoff Flows

In the flow in Figure 2 "Requirements Tree for Advanced Verification Techniques" the requirements arrive 3 potential sources:

- Often requirements are captured as tickets in a change management tool (such as Jira).
- They can be manually entered.
- Or they may from a larger ecosystem including requirements database systems such as Doors or Integrity.

Within the requirement mapping process the following steps are followed

- The requirement quality gateway maybe ensured (if they have not already been ensured during requirement entry).
- Requirements are mapped to features, sub-features, verification goals and their associated metrics. For example functional coverage points, code coverage, assertions, formal properties, directed tests, constrained random tests, code reviews, manual signoff, etc.
- A review of requirements between design specification and associated verification plan to ensure common understanding of requirements and planned implementation. This will ensure review for completeness of the design and verification plan.
- Verification execution results are imported and analysed against the verification goals contained in the plan to ensure requirements coverage. This is performed throughout the verification execution phase so progress can be tracked. User defined pass criteria can be used as interim milestones to aid progress tracking.

Finally requirements proofs may be exported in an XML format for use within a wider requirements engineering ecosystem (such as reqtify) or in PDF documentation format for full reviews and audit.

The XML export may conform to industry standard formats for import into other applications. Various XML formats have been found to be useful

- An internal format for data exchange with other projects or across the project hierarchy.

- User specific formats such as asureSIGN™ Requirement Qualification Engineering (ARQE) format (still evolving).

It is useful to support both a full or partial export of the requirement hierarchy, with the user selected details, to be imported into another project or to be used for reporting. Examples of user selected details might be

- Planning fields: Name, Description, Milestone, User, Kind, Type, Obsolete, Percent required, External ID, mapping criteria, Manual Sign Off, Aggregation, Parameter etc.
- Result fields: such as Bins, Hits, mapping, Kind, regression, pass/fail, etc.

Note that the database also needs to be able to extract data from the verification environment and this is done in 2 ways either by:

- A simple API which allows tests to report their status during their execution. For example, did the test compile, execute, pass or fail?
- The test can also tell the database the location of the log file for evidence used in subsequent audit trail generation.
- The ability to extract information from simulation or formal verification result databases.
- This has become easier with standardization of interfaces to these databases such as UCIS1)
- This allows automatic extraction of functional, assertion and code coverage, and formal verification results.

The rest of this section outlines the advantages of using a central SQL database solution.

5.3 Requirements -> test plan import via xml

The Requirements may be written in an external tool to a feature set level – this may then be reused as a top-level test-plan. Alternatively the feature level Requirements or top-level test plan may be written directly into asureSIGN™. This brings the requirements to the verification and test engineers, thus ensuring that they have the same comprehension of the requirements as the concept or design engineers.

This also allows for earlier test/verification planning, the design document will still be required for a full refinement of requirements into goals (bottom level test plan), but an earlier understanding can allow for understanding of test bench requirements and methodologies to be used etc. It also allows the database tool to sit within a larger ecosystem Product line management tool set.

5.4 Data Integrity, hierarchy, data translation

The biggest issue with the linking of multiple tools across multiple product variants is ensuring a common understanding of the requirements at each level. This includes ensuring preservation of the hierarchy as contextually requirements may change relating to their position within the hierarchy.

With asureSIGN™ this issue is avoided by preserving the hierarchy from the requirements database during import. The refinement of these feature level requirements (or top level testplan) is implemented within asureSIGN™ and thus avoids any corruption or mis-translation through the requirements traceability tree.

5.5 Change management – instant update

Any changes to the requirements can be automatically identified within the tool to ensure the change management reaches directly to the test and verification engineers. This alerts the user to identify any test updates, new tests to be written or redundant ones removed, in order to satisfy the new or rejected requirements with minimum effort.

5.6 Live database -> easy documentation

Usage of a live database in the same environment as the tests that are being managed allows for easy updates of the test descriptions should, during the project lifecycle, the tests themselves be updated due to change of requirements or technology improvements.

Often the test documentation is left until the end of the project with the information on any changes made during the project lost or forgotten. Poorly documented tests lead to poor and complex maintenance issues with the tests and test benches.

5.7 Tailored Documented proof

All fields within the database may be selected for export to PDF (or XML), allowing for documentation at a particular milestone either to goal level, mapped metric level or to result level – giving proof of requirements, how they were mapped into test and their results for audit purposes. They may also be chosen based on being obsolete or unmapped, therefore when changes are identified for a product variant you may extract a list of all the new tests/metrics required to be written and those that may be removed from the regression list.

5.8 Allows reviews of implementation document against test plan

The requirements may be translated via xslt into the introduction chapters of any design specification to indicate the requirement features required within the chapters as well as for the database (as a testplan). This allows for cross-referencing when the design specification is written and can identify some interesting issues:

- Over-engineering: If the test engineer reviews the design specification and finds a feature to test that they cannot relate back to a requirement, then this should go back through the change management process, as this may indicate "Over-engineering". For example, the design engineer has added a feature that is not necessary for the current product variant.

  Note that over-engineering can be deemed to be a positive thing in many cases but if it is not evaluated as part of the wider design, resources and other product restrictions then it may cause issues.
- Completeness: If an extra feature, as above, may actually be a required feature that has not been documented within the requirements list, this helps ensure completeness as does the test team not seeing design details of a requirement in the design spec. This helps ensure a complete design spec.
- Ambiguity: If the design team and test team disagree with how a requirement is interpreted then it needs to go back through the change management to clarify what the

requirement means and to ensure that it is rewritten to be unambiguous. This is also a requirements quality checkpoint.

5.9 Mapping:

Mapping ensures that the test domain is tightly coupled into the Requirements Engineering domain. asureSIGN™ supports multiple types of metrics: directed test, random tests, structural coverage types, functional coverage types, assertions, formal properties, manual sign-off – they can also be assigned as obsolete for particular products. The tool allows mapping from the goals (test descriptions) to the metrics types through a choice of available metrics of the selected types.

asureSIGN™ also allows the user to grade a test and match the grading on selected milestones. The milestones can be tailored to match the project management. Allowing visibility of the tests and grading allows for early releases of IP Modules into sub systems and full systems; this helps ensure communication between the groups and stops over verification (multiple debug of the same bugs). For example, a first release may just test the interfaces to allow early integration of an IP Module, when the IP Module is integrated into a larger system the engineers are aware not to test or debug any functionality outside of the interfaces for integration. It also allows for the user to define what % of sub features and goals need to passing at a particular milestone to understand that the project is progressing in a timely manner.

5.10 Test management:

Supporting a round robin solution which allows the user to export and import the database information via XML with control, has a couple of advantages:

1. Use of this is to allow the user to do bulk-updates outside of the tool for ease of use.
2. It may also be used for allowing reuse of the requirements with their mapping between product variants.

The tool indicates what the changes are when a partial import occurs on an existing database, and at this point it is able to identify and indicate the new and now obsolete requirements so users are aware of the new metrics required to be implemented and the ones that are not required to be maintained or implemented for this particular variant.

Due to the variety of types of metrics that may be supported, some have a simple pass/fail criteria and some have a more complex analysis required to indicate whether or not a metric is working correctly. This solution allows the user the ability to control how results information is analysed and what the success indicators are.

Allowing the user to look across two or three selected regressions helps users to analyse trends, project progress and areas of concern easily.

5.11 ISO26262 compliance

ISO26262 mandates a hierarchical Requirements Engineering approach, any solution must have a clear Hierarchy window and also allows imports of multiple databases into others to build bigger level hierarchies if required to ensure this exists throughout the entire lifecycle of the project.

ISO26262 requires change and configuration management – any tooling must integrate with any version control system and be managed to support automated change management support, or implement its own change and configuration management solution.

Security must allow protection of data at all times, so supports of baselining for future audit proofing and the ability to replicate earlier database data is essential for such a tool.

5.12 Compliance / Audit Management

Documentation for audits is held within the tool and is readily available to incorporate into deliverables

Compliance to different industry standards must also be considered for any such tooling solution

6. Advantages of RDVT

RDVT offers many advantages to Requirements, Verification and Project management and those working on Compliance and Audit.

6.1 Requirements Management

Using requirements within the test flow allows test engineers early analysis for planning and a ‘shift-left’ approach for quality improvement. A requirement driven test methodology then assists with achieving a complete and high quality set of requirements, through enforcing a well-managed process. Finally, the mapping of requirements through tests to results ensures requirements engineering completion.

6.2 Verification Management

Understanding the verification status in terms of externally focused customer requirements rather than internal metrics, allows users to have far more controllability over their projects and assists with ensuring early releases, with better communication of the status of the early releases. It should also be possible for a user to generate partial reports based on their particular set of requirements or interests (such as a block within the hierarchy, software requirements, power management requirements, etc.).

6.3 Project Management

Reusability of data across projects allows for reduced ramp up times is essential for state of the art products – re-implementing mappings is a waste of resources and also a risk to data integrity. Variant management across projects helps to easily identify redundant tests and new tests needed and is essentially to ensue well maintained tests and test benches

Early verification efforts during the planning phases (such as identifying tests, coverage, assertions and linking them to features) are difficult to measure. This can lead to a lack of visibility of the progress during these stages. Introducing suitable metrics and tracking them leads to better early stage project management by making it easy to identify areas that need attention

Milestone grading assists with resource management and allows for improved communication across the project, most requirements management tools do not allow good management of the milestone process, this was identified by Infineon as being a tooling gap.

Enabling risk-based testing (e.g. adapting the level of testing according to risk or SIL assignment) helps manage the importance and cost of tests and their failures. Combining data from multiple tools into one system or combining data from multiple projects into one underlying database are also necessary to help manage and reuse tests, test benches, mapping and requirement data.

The visibility of requirements mapping to tests allows for a secondary requirement quality check and a process to help ensure completeness of the requirements, which is also deemed to be a major failure point within requirement engineering.

6.4 Impact Analysis

Requirements engineering tools, do allow for attributes to allow manual cross linkage to be implemented, this is often extremely resource heavy and very difficult to implement, examples such as puresytems:variants and Biglever Gears are two such solutions that aim to solve this whilst also investigating impact into different product variants.

The mapping of the tests into the requirements does offer the user a quick analysis of how much test/verification resources will be impacted if a change is implemented, attaching attributes to the tests to ascertain their 'grading/expense' may also assist the impact of requirement change.

TVS is currently also investigating using more automation within the impact analysis to reduce the amount of manual input. Due to the requirements and test being managed within the same tool there is the possibility to use some intelligent algorithmic solutions within the tool to assist with this.

6.5 Product Line Engineering

Product Line Engineering is about using methods, tools and techniques for creating a collection of similar systems, from a shared set of assets, using common means. Whilst primarily used in software, it is becoming increasingly used in the semiconductor hardware industry due to the drive into stricter requirements engineering practices mandated by new and improved standards, and also market pressure for more tailored product solutions and thus product families as opposed to historical single products.

6.6 Variant management

The variation issue is not limited to the requirements themselves, but permeates throughout the product life cycle from inception, requirements, design, implementation to test and their results and across multiple domains in the hardware world such as pre-silicon IP/Module(unit), subsystem and SOC (System-on-chip), to post-silicon IP/Module and SOC validation, through to firmware, test, analog and Software. All requirements on the multiple variants may be tested at all, multiple, or none of these domains and may contain multiple dependencies. This makes the variant management of the data a truly orthogonal problem requiring a strict process and tool flow to ensure correct implementation.

With asureSIGN™ we considered the variant problem from the experiences within the pre-silicon IP verification of the Infineon Aurix™ automotive microprocessor product family, which consists of 5 products. The SOC (system on chip)is made up of multiple IP Modules (these are the equivalent to software units), in this case between 60-80, all of which are tested by multiple teams with intricate and complex testbenches across multiple worldwide sites. These IP Modules are released into subsystems which allows the gluing together of systems within the same site as the IP development team, to allow for use of the local specialist knowledge during this process. The sub-system is then shipped to another site in another country to be put into the SOC. There are approximately 3000+ external requirements alone on the product family and multiple stakeholders.

7. Conclusions

In this paper we have introduced a Requirements Driven Verification and Test (RDVT) methodology and explained how this methodology can be used to support compliance to various hardware (and software) development standards. However, more than this we have also shown that advanced verification techniques can be deployed in RDVT. We explained the advantages found in using an SQL database approach to capture the requirements tree and mapping to the metrics typically generated by advanced verification techniques.

Infineon have accepted the asureSIGN solution, developed within the CRYSTAL ARTEMIS project as state of the art and has rolled it out currently within its Automotive Microcontroller group, it is currently being rolled out to the post silicon group and expanding into further Infineon groups. Ongoing tool improvement both with the CRYSTAL project and externally within the Rail and Avionics domain to expand and improve the tool to ensure it has the ability to support all tooling solutions regardless of domain.

8. CRYSTAL (CRitical sYSTem engineering)[2)]

The overall project goal of CRYSTAL is to foster Europe's leading edge position in the design, development, and deployment of interoperable safety-critical embedded systems in particular regarding quality, cost effectiveness, flexibility, reusability, acceleration of time to market, continuous integration of innovations, and sustainability.

asureSIGN™ was developed by TVS[3)] for Infineon's[4)] AURIX™ [5)] 32-bit multicore Tricore product family in order to achieve implementation of interoperability within the tooling as part of the CRYSTAL project. The goal of the CRYSTAL project is to provide a platform of interoperability between tools supporting all the steps constituting the lifecycle of a product.

"The work undertaken leading to these results has received funding from the European Union's Seventh Framework Program (FP7/2007-2013) for CRYSTAL – Critical System Engineering Acceleration Joint Undertaking under grant agreement № 332830 and from the UK's TSB[6)] Technology Strategy Board"

Further information from the CRYSTAL project can be found in "Crystal 3rd European Conference on Interoperability for Embedded System Development Environments" [2)]

Further information on asureSIGN™ can be found at [3)]

9. References

[1)] UCIS http://www.accellera.org/activities/committees/ucis/

[2)] CRYSTAL http://www.crystal-artemis.eu/

[3)] TVS http://www.testandverification.com/

[4)] Infineon http://www.infineon.com/

[5)] AURIX™ http://www.infineon.com/cms/en/product/channel.html?channel=db3a30433727a44301372b2eefbb48d9&ic=0101033

[6)] TSB

[7)] ISO 26262-1:2011 Road vehicles -- Functional safety

10. Appendix A: Sample Hierarchy

Figure 3 "Requirements Signoff Flow for Advanced Verification" shows a sample hierarchy from a project. It also shows the software requirements as well as interface and hardware requirements.

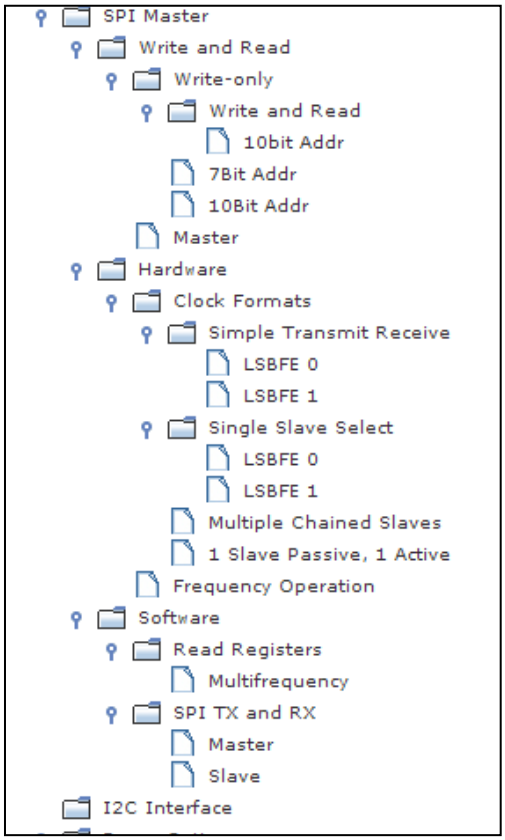

Figure 4: Sample Hierarchy