Assurance Plan for Complex Electronics:

Assurance Process: Preliminary Design

This printer-friendly page contains the following sections:

Preliminary Design Phase

This Overview page for the Preliminary Design Phase contains the following sections:

Overview

The preliminary design phase may also be known as *conceptual design* or *architectural design.* During this phase, the high-level design concept is creates, which will implement the complex electronics requirements. This design concept may be expressed as functional block diagrams, design and architecture descriptions, sketches, and/or behavioral HDL (hardware description language).

The objective of the design phases (preliminary and detailed) is to create a design that will correctly and completely implement the requirements. For the preliminary phase, the main goal is to map out how the complex electronics will perform the functions specified in the requirements, within the constraints of the device, the defined interfaces, and the environment the device will operate within. At this phase, the designer needs to maintain a systems perspective and look at the complex electronics operations in concert with the rest of the system. Now is the time to identify inconsistencies, misunderstandings, and ambiguities.

The objective of design assurance is to verify that the design does implement all the requirements, and that it implements nothing but the requirements. Any deviations (such as additional functionality that may indicate a missed requirement) are fed back to the requirements engineering process.

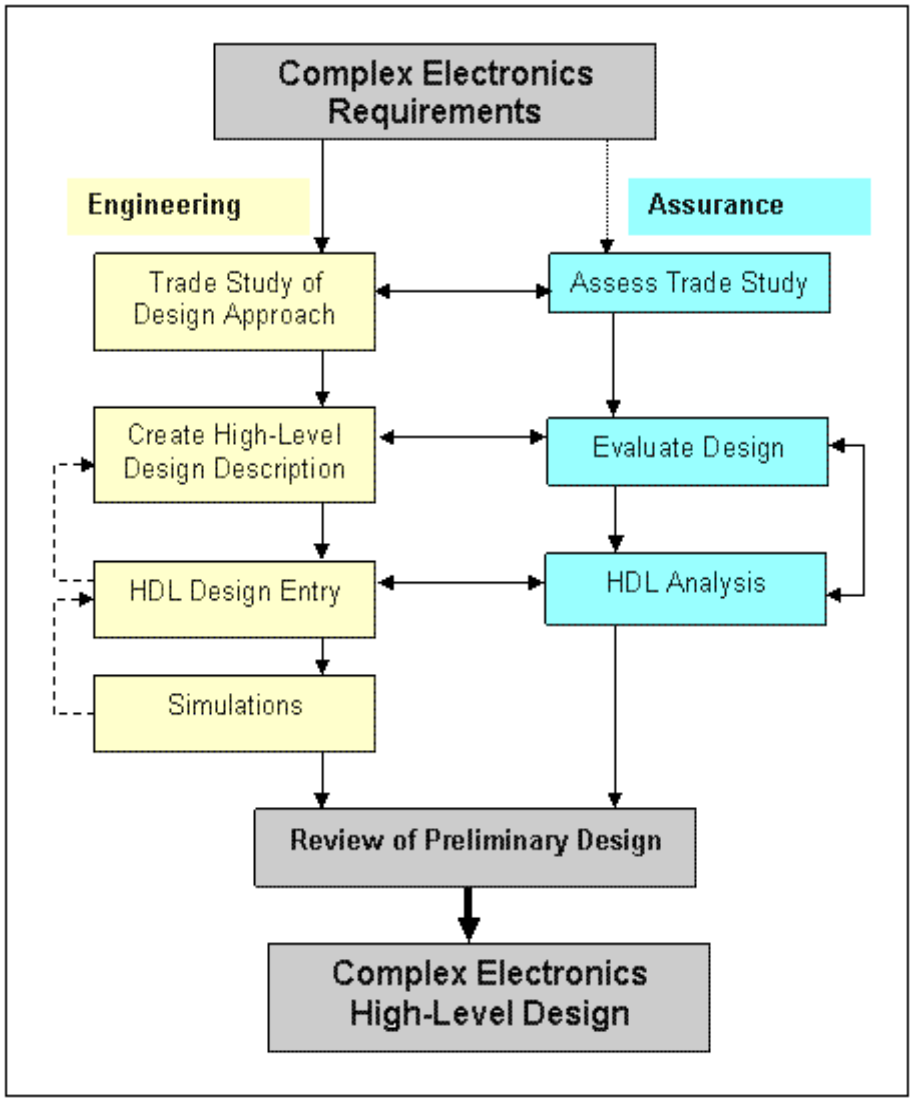
The main design activities for the preliminary design phase are:

1. Create the high-level design description.
2. Identify major components, including third-party IP modules or cores.
3. Any derived requirements that result from the process are fed back to the requirements engineering process
4. Any omissions or errors are resolved
5. Include reliability, maintenance, and test features that are necessary to meet performance and quality requirements, and to ensure that testing can be performed to verify the requirements.
6. Identify constraints on other system elements that are a result of this high-level design.

This assurance process for complex electronics assumes that complex electronics requirements have been developed, assessed, and baselined (formalized). In reality, these requirements may be included in a higher-level assembly requirements specification, such as a circuit board or sub-system. The requirements for complex electronics are likely to be a work in progress, as design decisions in other areas of the system influence the functions the CE device will perform. Requirements management will be an important process throughout the design, implementation, and test phases.

Preliminary Design Process

The diagram below shows the preliminary design process for complex electronics. The Develop Preliminary Design page describes the engineering process to create the design. The Assurance Process page describes the activities to assess and verify the design.



Entrance Criteria:

The following criteria should be met prior to beginning the development and assurance of the complex electronics preliminary (architectural) design.

- The complex electronics requirements should be complete and baselined. In reality, the requirements must be sufficiently complete to allow the design to be started.
- The Configuration Management process is defined and in operation.

- The Assurance process is defined and operational. The assurance engineers responsible for the complex electronics have been identified.
- The hardware description language (HDL) and toolset have been selected. HDL coding guidelines for the project are complete.
- Design guidelines are specified.

Exit Criteria:

At the end of the preliminary design phase, the following criteria should be met:

- Requirements for complex electronics are complete, with only limited exceptions.
- The design architecture is approved.
- Behavioral simulation has been performed (if required)
- Models and test benches required as inputs to the Detailed Design phase (e.g., synthesizable RTL models) are generated.

Roles and Responsibilities

The table below describes typical activities during the Preliminary Design Phase for both engineers and assurance personnel.

Preliminary Design Phase	
Role	**Typical Activities**
Systems Engineer	Ensure that design decisions at the sub-system level (including the complex electronics) are consistent with the overall system design. Manage the interfaces between the various system elements. Ensure that the sub-system designs will be verifiable. Start system integration planning.
Electronics Designer	Assess the complex electronics architecture for compatibility with interfacing electronics. Plan for integration of the complex electronics with supporting electronics. Ensure the CE is testable in-circuit.
CE specialist (optional)	Create the architectural design for the complex electronics. Create the test bench to evaluate the design. Perform behavioral simulations. Work with the systems engineer to resolve interface issues.
System Safety	If the complex electronics is safety related, review the architectural design for proper implementation of safety features. Ensure that the CE does not create any new hazards.
CE/Quality Assurance	Assess the complex electronics design against the requirements. Perform other analyses required by the CE classification. Ensure that the documented design process is in place and is followed by the design team.

Preliminary Design Site Map

The table below describes the information contained on the other pages in the Preliminary Design section.

Preliminary Design Site Map	
Overview	This page
Develop Preliminary Design	The process of developing the architectural design for complex electronics.
Assurance Process	Describes the assurance process for complex electronics preliminary design, with links to specific techniques.

Develop the Preliminary Design

The role of the complex electronics engineer during the preliminary design phase (also called the *conceptual* or *architectural* phase) is to come up with a high-level design that meets the requirements. At this level, the design engineer has to keep a systems point of view, because the complex electronics interfaces with other elements of the system. It's also common that the functions to be performed are still fluid, and the allocation to software, complex electronics, and other electronic elements may change.

The architectural (high-level) design converts the requirements into functions to be performed and the interfaces between those functional blocks. There is often a certain amount of trade-off during this phase, as various requirements or goals come into conflict. The design engineer needs to be aware when design decisions for the complex electronics affect other elements of the system, especially by imposing constraints on those elements.

The initial high-level design may be made using functional block diagrams, design and architecture descriptions, or sketches. As the design is solidified, it is usually implemented in a hardware description language (HDL), such as VHDL, Verilog, or SystemC. At this stage of development, the complex electronics is defined as black boxes, with specified behavior and interfaces. The specific details are added later in the design life cycle.

During the preliminary design phase, the architecture of the chip is defined, verified and documented. The architecture is defined down to the level of basic blocks that implement all intended functions. In addition, the interfaces and interactions between the blocks are specified. Important decisions for the implementation of the chip are made at this phase, including the selection of COTS, third-party, or re-used IP (Intellectual Property) modules or cores.

This section covers:

Design Description

The design description (architecture) is the preliminary design document. The design description should show a clear relationship between the requirements (specification) and the design. A traceability table is one way to accomplish this.

The document provides an overview description of the functions the complex electronics will perform, and includes:

- An introduction to the architecture
- Partitioning of the design into blocks, with supporting rationale
- Interactions between the blocks
- Internal protocols and data formats
- Interfaces and protocols to external devices
- Memory mapping
- Detailed functionality (e.g. finite state machines)
- Fault/error handling

The high-level design needs to address the following areas, in addition to meeting the requirements:

- Constraints related to safety, including prevention of design errors
- Handling of functional, component, over-stress, reliability and robustness defects
- Constraints the design imposes on other system elements, such as software

- Reliability
- Maintenance
- Test features – what needs to be included to make sure the device is testable

Create HDL Models

Most modern designs are implemented in a hardware description language (HDL). HDLs are any languages that are used for formal description of electronic circuits. These languages can describe the operation, design, and simulation tests of the circuit. HDLs can show several aspects of the design, including the temporal behavior and spatial structure. The two primary hardware description languages are VHDL and Verilog.

Using an HDL, the designer creates a model of the complex electronics that includes the functions (behavior) and blocks of the architectural design. Later in the design phase, the HDL model will be synthesized (essentially compiled) by the tool suite into lower-level models. HDL models can be used to simulate the behavior of the complex electronics through execution in a test bench (described below) or simulation tool. Simulation allows for early verification of critical functions, though timing behavior is usually not verified at this phase of design.

HDL models have two aspects, behavioral and structural. A behavioral specification defines the behavior of a digital system (module) using traditional programming language constructs (e. g., ifs, and assignment statements). This description of a complex electronic device divides the device (chip) into several functional blocks that are interconnected. HDL is used to describe the behavior of each block. Functional blocks can be a finite state machine, a set of registers and transfer functions, or even a set of other interconnected functional blocks.

A structural specification describes the digital system (module) as a hierarchical interconnection of sub-modules. The components at the bottom of the hierarchy are either primitives or are specified behaviorally. It is in the structural specification that individual inputs and outputs are defined.

Hardware description languages can be used to describe complex electronics at many different levels of abstraction. An abstraction is a simplified representation of something that is potentially quite complex. It is often not necessary to know the exact details of how something works, is represented or is implemented, because we can still make use of it in its simplified form.

The levels of abstraction for a complex electronic device are:

- System or Behavioral
- Algorithm
- Register-Transfer Level (RTL)
- Gate

Preliminary (architectural) design goes down through the algorithm level. At this level, there is no concept of *timing* in the description of the complex electronics behavior. A pure algorithm consists of a set of instructions that are executed in sequence to perform some task. A pure algorithm has neither a clock nor detailed delays. Some aspects of timing can be inferred from the partial ordering of operations within the algorithm. The algorithmic level of abstraction is similar to software programming ("While ready, do task A and task B, then do task C"). Because of the lack of timing information, this level is not synthesizable (able to be mapped to hardware).

The RTL level models are created during the detailed design phase, and will be described there.

Test Benches

A Test Bench is a HDL design you create which can load your circuit, apply stimulus to its inputs (including defining multiple clocks) and check the outputs for correctness. Because it is a program you write, you have control over how your circuit is built and simulated. In addition to the above capabilities, a test bench can

provide behavioral or structural models for everything on the PC board. In this way, it enables you to simulate the entire system including your complex electronics design(s) as well as external bus interfaces, external memories, etc. An engineer can design the test benches to automatically check important data conditions and to report any errors to a command window.

Comprehensive, up-front verification is critical to the success of a design project, and test benches should be created as you start to design your device. A HDL test bench/simulator can become your primary design development tool. When simulation is used right at the start of the project, you will have a much easier time with synthesis, and you will spend far less time re-running time-intensive processes, such as place-and-route tools and other synthesis-related software.

Test benches can be quite simple, applying a sequence of inputs to the circuit over time. They can also be quite complex, perhaps even reading test data from a disk file and writing test results to the screen and to a report file. A comprehensive test bench can, in fact, be more complex and lengthy (and take longer to develop) than the circuit being tested.

Depending on your needs (and whether timing information related to your target device technology is available), you may develop one or more test benches to:

- Verify the design functionally (with no delays).
- Check your assumptions about timing relationships (using estimates or unit delays).
- Simulate with annotated post-route timing information so you can verify that your circuit will operate in-system at speed.

A typical VHDL or Verilog test bench is composed of three main elements:

- Stimulus Generator - drives the unit under test with certain signal conditions (correct and incorrect values, minimum and maximum delays, fault conditions, etc.)
- Unit Under Test - represents the device undergoing test or verification
- Verifier - automatically checks and reports any errors encountered during the simulation run. Compares model responses with the expected results.

Coding Standards

A coding standard is important when more than one person will ever have to use the source code. The big danger is that when the person who wrote the original code leaves or moves on to another project, no one will understand how it works if the code ever has to change. Even the original designer is likely to forget it in several months.

One can easily write individual lines of understandable HDL code that collectively become extremely difficult to follow. A good coding standard will help alleviate this by providing guidelines for hierarchical structures and component instantiations. For instance, many books use various types of flip-flops as examples to model component instantiations (mostly because these are already understood by the readers). However, in practice, it's generally poor coding style to instantiate logic by mapping each register to various kinds of flip-flops. This can lead to longer, more obfuscating logic that does not take advantage of the ability to write in VHDL and Verilog at a higher level.

The important factors in a HDL coding standard are:

- Consistent and defined style
- Guidelines on writing understandable code
- Commenting guidelines
- Information to capture in comments at each level
- Naming convention (for consistency)

Design Verification

Once the preliminary design and HDL model are complete, the designer usually performs some level of verification using simulation. A test bench allows for software simulation of the complex electronics, and testing of the behavior of the design under various inputs. Software simulation is a very common verification and test tool for complex electronics. Other approaches to design verification include:

- Hardware accelerated simulation uses special purpose hardware to accelerate circuit functions, allowing long simulations to be run in a short period of time.
- Emulation uses electronic components to emulate the circuit behavior (e.g., using FPGAs to verify an ASIC design).
- Rapid prototyping creates a physical model (prototype) of actual hardware.
- Formal verification, using model checking to verify properties relative to a model (formal specification) and/or theorem proving, which proves theorems regarding properties of the mode.

While simulation is a very useful tool, it requires careful planning to generate appropriate inputs (stimuli) for the complex electronics and to ensure that the design is thoroughly tested. Simulation does not take the place of in-circuit testing, but can be used to perform low-level verifications of design properties. Simulation can also be used to explore the design's responses under various error or fault conditions.

Maintenance and Maintainability

If the device will potentially need to be maintained (including reprogramming updates), this issue should be considered early in the design of the complex electronics and its supporting circuitry. Some areas to consider are:

- Will the device architecture allow for the types of enhancements that can be foreseen?
- Does the design specification provide the information that an engineer would need to understand how the product works?
- Is the HDL code readable?
- Are comments liberal and informative?
- Is the necessary physical infrastructure in place to allow reprogramming?
- Is access to the reprogramming port, if one is used, available when the system is installed?

Preliminary design review

This review is not the project Preliminary Design Review (PDR), but a review and evaluation of the complex electronic preliminary design. They should be reviewed by a variety of people. Each person will bring a viewpoint to the review that can identify gaps or errors in the design, provide *best practices* for design elements, and ensure that all requirements are implemented in the design. The preliminary design review can be a formal review (such as part of the project Preliminary or Critical Design Review), an informal review, or the result of many individual reviews.

Details on Design Reviews can be found in the Techniques section.

Assurance Activities for Complex Electronics Preliminary Design

As the engineers develop the high-level design for complex electronics, the assurance engineer(s) works alongside to assess the processes and the product (preliminary design). These two activity streams are opposite sides of the same coin, and result in a verified design.

Use the Tailoring chart to determine which activities or analyses are required for a particular assurance classification. Activities that are not required may still be performed, if desired. The assurance activities for complex electronic preliminary design include:

- Preliminary Design Evaluation
- Preliminary Design Review (formal or informal)
- Simulation
- Process Verification
- Updating:
  - Risk analysis
  - Interface Analysis
  - Traceability Analysis
  - Criticality Mapping
  - Fault Tree Analysis
  - Failure Modes and Effects Analysis

Every method listed above does not have to be applied to every project. The table below uses the Complex Electronics Classification to map the activities, and depth of each activity, against the classification. This table allows for easy tailoring of the assurance activities to the device complexity and assurance classification.

Tailoring Guidance for Assurance Activities - Preliminary Design Phase			
	Low	Moderate	High
Preliminary Design Evaluation	Evaluate design for compatibility with rest of system	Evaluate for compatibility, interfaces. Have expert evaluate important design areas.	Use expert to evaluate design
Preliminary Design Review (formal or informal)	Informal	Informal or Formal	Formal, with engineering expert
Simulation	Review designer's simulations and tests	Review designer's simulations. Use designer's test bench for simulations of important design elements.	Perform completely independent simulations
Process Verification	Informal	Moderately formal	Formal Audits
Risk analysis	Informal	Informal	Formal
Interface Analysis	Informal assessment	Focus on key interfaces and critical timing	Detailed analysis, all interfaces. Includes timing
Traceability Analysis	Trace from complex electronics requirements to design blocks	Trace from complex electronics requirements to design blocks. Verify blocks correctly implement the requirements.	Trace from complex electronics requirements to design blocks and the reverse. Verify no functionality not covered by requirements.
Criticality Mapping	Not Performed	Map critical functions to complex electronics architectural blocks.	Map critical functions to complex electronics architectural blocks.
Fault Tree Analysis	Not performed	If FTA exists, map appropriate failures to CE architectural blocks.	If FTA exists, map appropriate failures to CE architectural blocks.
Failure Modes	Not performed	If FMEA exists, re-evaluate it for	If FMEA exists, add appropriate failures

and Effects Analysis		possible effects of failures at the CE architectural block level.	of CE architectural blocks and trace to system impacts.

Preliminary Design Evaluation

A key assurance activity for any project is expert evaluation and assessment of the design. For complex electronics, some portions of the design can be evaluated by an assurance engineer with sufficient understanding of complex electronics. Most assurance engineers will not be able to provide expert assessment, however. If a design is safety or mission critical, an independent engineer with experience in complex electronics should provide a detailed evaluation of the design.

For complex electronics, the questions to ask when evaluating the preliminary design are:

- Is the design is complete (i.e., are all the requirements implemented)? Do all requirements trace to architectural blocks or functions?
- Are the requirements correctly implemented? If the requirements were ambiguous, various people will have different interpretations of what the device should do.
- Does the design contain functions that are not required? If so, this may indicate missed requirements.
- Does the design contain any internal inconsistencies or conflicts?
- Does the design conflict with any requirements or with other system element?
- Does the design clearly identify constraints on other system elements, and on the installation and operation processes?
- Is the quality of the design (documentation and HDL model) adequate for its usage? Consider if the design is readable, understandable, and maintainable.
- Will the design result in a testable system? Consider both physical access (e.g., test pins) and the ability to test modules within the device.
- Are special pins (e.g., mode pin on FPGA, JTAG pins, no-connect pins) used correctly?
- If some requirements conflict (such as power consumption and performance speed), was a trade-off performed to determine the optimal levels for each requirement?
- Is the design flexible enough to accommodate anticipated requirements changes?
- Did the design follow the coding standard and design guidelines?

If COTS or re-used IP cores or modules are used within the complex electronics, they need to be evaluated for how they will operate within and interface to elements of the complex electronics. Areas to consider are:

- Functions within the IP module that will not be used.
- Interfaces (voltages, timing, signal characteristics, etc.)
- Verification performed by the vendor. What documentation exists for that verification? What additional testing needs to be performed?
- IP module upgrades. If the vendor indicates that there is a problem with the IP module, how will that problem (and fix) be evaluated for the complex electronics?

A preliminary design evaluation checklist can be found in the Checklist section.

Preliminary (Architectural) design review

A Design Review is a more formalized process that brings together individuals with various understandings of the system and the complex electronics. The goal is to have each person review the design (preliminary in this case) against their area of knowledge. By performing a wide review, problems related to interfaces, system operations, etc. are more likely to be found.

During a design review, the reviewers need to verify that:

- All requirements are addressed, including environmental, safety, and reliability
- Safety aspects of the design are explicitly identified
- Design can be implemented, tested, and maintained
- Any new manufacturing or other techniques are evaluated
- Design is traceable to requirements

Details of design reviews, including a discussion of reviewing HDL code, can be found in the Techniques section.

Simulation

The assurance engineer has two simulation tasks at the preliminary design phase: 1) evaluate the designer's simulations and 2) perform independent simulations. For complex electronics classified as low assurance, independent simulations are not necessary. The amount of independent simulation will depend on the CE classification and the functions the CE will perform.

The designer's simulations, including test benches, need to be evaluated for correctness. Test benches are created by human beings, often by the designer, and are subject to faults and failings like any human endeavor. If the logic of the test bench is incorrect, or if a particular stimulus is not defined, then the end result of the tests may not show an actual error. You can't assume that the test bench accurately and completely tested the device - especially if the device will be used in a safety-critical application. For the simulations (tests) that the designer performed, consider:

- Does the simulation show that the architecture will meet the requirements?
- Does the simulation address all expected inputs and combinations of inputs?
- Does the simulation consider inputs that are out-of-range or otherwise invalid?
- Does the simulation include the effects of faults or failures in other parts of the system?

If an independent simulation is performed, the first question is: can I use the designer's test bench? For CE classified as moderate assurance, that is probably alright. Using an already developed test bench will save a lot of time an effort. Just be aware that there may be errors in the test bench, as well as in the complex electronics model under test. For high assurance devices, the test bench should be developed by the assurance engineer or the person who will perform independent simulation.

The questions asked above are valid for an independent simulation. You want to design tests that will show if the design does, or does not, function properly in all anticipated situations. Designers are success oriented, so you should focus on unexpected inputs, sequence of inputs, or simultaneous combinations of inputs. Simulations can focus on small modules or parts of modules, as well as on the complex electronics model as a whole.

Creating a simulation involves a thorough understanding of the HDL used in developing the test bench and the methods of testing complex electronics. It is usually much easier to learn to review a simulation than to create one from scratch. The level of expertise of the assurance engineer should not preclude independent simulation, if that is called for. An engineer familiar with simulations, but not part of the project, can be used to develop the test bench and test cases, and possibly to even execute the tests.

Process Verification

Process verification is an activity performed at each life cycle phase. The plans that were generated at the beginning of the project should be followed as the complex electronics is developed. Assuring that the processes are followed is one way to help the project stay on track. Processes are not written in stone, but can be changed to reflect project reality. One result of process verification is to advise the project when updating the processes is a necessary activity.

For the preliminary design phase, the assurance engineer will:

- Verify that the entrance criteria were met before the project moved into this phase. If not, document the increased risk due to non-approved requirements or other issues. Provide risk mitigation suggestions to the project.
- Verify that the exit criteria are met before the project moves to the Detailed Design phase. If not, document the increased risk due to non-approved requirements or other issues. Provide risk mitigation suggestions to the project.
- Check that the Design Description, together with the documentation of previous development phases, is complete and documented in a level of detail sufficient to proceed with the Detailed Design.
- Review selected tools for applicability to the design process. Check the tool vendor web-site and other sources for known tool defects or operational workarounds.
- Make sure a disciplined design process is in place, and the design engineer is willing to follow it. Negotiate as necessary.
- Check that the planned measures, tools, methods and procedures have been applied.
- Ensure that the complex electronics preliminary design is under configuration control. At the end of the phase, ensure that the preliminary design is approved and baselined.
- Ensure that the design team is using a design and coding standard. This standard will define the basic design philosophy and specify aspects of the HDL program structure. Even if only one engineer is designing the device, a standard 1) helps ensure that the HDL program is understandable by others (and the design engineer, six months down the road) and 2) provides a way to capture and incorporate best practices in the design process.
- Verify that all planned activities for complex electronics were performed.

Another process verification activity at the beginning of this phase is to review the design guidelines and coding standard (after ensuring that the project has a coding standard). A design and coding standard should include:

- Specific HDL coding features and methods that either should be used, or should not be used.
- Design “best practices”, either as a guideline or as requirements.
- Naming conventions for modules, inputs, outputs, etc.
- Commenting rules that define what types of information to include in comments. One example would be to define a module header that includes comments on the module's purpose and structure.
- Readability rules may be covered under naming and commenting conventions. But the standard should help guide the designer into creating HDL code that is readable by others.
- Modularization guidelines provide information on how to decompose the high level design into individual modules.

Update Analyses

Analyses performed during the requirements phase should be updated at this time.

Risk Analysis

Evaluate previous risks to identify those that no longer apply or that have changed their priority based on changes in probability or impact. Identify any new risks relevant to this phase of development and determine which require mitigation plans. Check that preventive measures and/or contingency plans exist for all identified risk items and that the risk, with mitigations in place, is acceptable for moving to the Detailed Design phase.

Design Interface Analysis

Are all interfaces identified, both internally and externally? Are they correct, consistent, complete, and accurate?

Traceability Analysis

Trace the requirements into the design elements and functions. Are all requirements implemented? Is there any additional functionality in the design that is not included in a requirement? Also, evaluate the tracing for accuracy, completeness, consistency, and correctness

Criticality Mapping

If a criticality map was generated at the requirements phase, expand the map down to the architectural blocks that implement the critical functions.

Fault Tree Analysis

If a Fault Tree Analysis was developed, and it traced to the complex electronics, expand the fault tree to include the appropriate architectural blocks. Also review the FTA against the preliminary design and ensure that the design incorporates sufficient fault prevention.

Failure Modes and Effects Analysis

If a Failure Modes and Effects Analysis (FMEA) was developed, and it included complex electronics, look at the failure modes defined for the CE device. Does the preliminary design provide any additional possible failure modes? If so, add them to the analysis. Consider failures in interactions between the complex electronics and system elements it interfaces with, including noisy signals and invalid outputs.