Best Practices in Verification Planning

Benjamin Ehlers - Verification Engineer, Freescale Semiconductor
Carmen Vargas - Verification Engineer, Freescale Semiconductor
Paul Carzola – Architect, Cadence Design Systems

Abstract

Creating meaningful verification plans is an art form yet to be fully codified. Failure to plan means planning to fail. The development of verification plans is the most crucial step in the functional verification process. Oddly, not every company takes the time to establish an in-house prescriptive and repeatable approach. This paper articulates a codified process for verification planning based on actual experience at Freescale Semiconductor. The verification planning process described in this paper is streamlined for derivatives and compressive enough for new design and verification development. It explains a complete verification flow including verification strategy, planning, change management and closure.

I. Introduction

Let's face it, effective planning is not a perfectly easy process that every engineer naturally understands or enjoys. Yet, effective verification planning can be quite beneficial in providing a significant return on investment in the quality and efficiency of the resulting verification product. By starting with a culture of methodology driven verification, an acceptance that effective planning must span the entire verification flow, and by following some fairly simple structured and well organized practices, the process of verification planning can be made effective, easy, and efficient and generate a high quality result. One of the more exciting elements of best verification planning practices discussed in this paper is in the development of a codified and executable verification plan. Some other best practice elements of the larger verification planning methodology presented in this paper may seem less obvious or even unrelated, but when considered with respect to a complete verification planning methodology should not be quickly disregarded.

II. Problems and Challenges

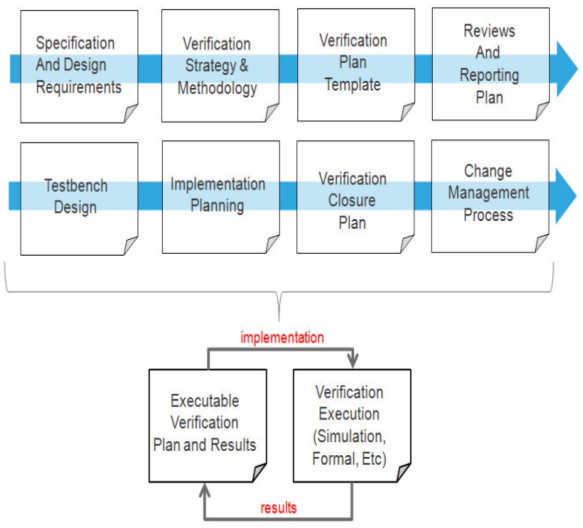
There are numerous barriers and challenges that exist counter to effective verification planning, many of which can be traced to the overall approach and culture of functional verification methodologies and strategies within a company or group. Human and technical factors contribute to the problems creating barriers to planning. One significant barrier to verification planning is a cultural issue, where some verification engineers would like to treat verification planning as an afterthought if it is done at all. This often leads to an ad hoc approach to the planning process, little to no consistency among verification documentation, and a lower sense as to the certainty of closure and completion. Another barrier to verification planning comes in the form of those who recognize the value of verification planning but do not have an established and well organized planning flow. This tends to result in the verification engineers fighting a battle of inconsistency and incoherency in the structure and content of the verification plans and verification documentation. Additionally, a barrier exists for those who recognize the need for verification planning, but who also believe the notion that there is a large unrecoverable cost associated with spending time in proper planning due to the required time investment. Many of the barriers to planning at a company can be attributed in some manner to an absence of verification methodology adoption, gaps in the company's established verification methodology, and/or gaps in existing technology.

Some of the many challenges faced during verification planning which can interfere with or even nullify the effectiveness of the verification plan include the need for a large number of verification engineers working in parallel on a given project, the level of quality of design documentation and requirements, undefined or under-defined design blocks, frequent and significant change requests, and late design changes. Unless a verification project is a small block level design in a standalone environment, chances are that the verification done on any given project uses a distributed approach where many different verification engineers are working in parallel on the project and need the ability to bring together all of their individual plans and work together. This distributed approach presents a challenge to effective planning regarding the consistency and coherency of verification plans and verification documentation, which directly affects the ease of which the verification results can coalesce and be reported in a clear fashion. Another significant challenge in the verification planning process involves the quality of the design specification documents and the set of design requirements. Design specifications and design requirements are the primary input source needed for effective verification planning. The higher the quality of the design documentation is at the start of the verification effort, there is less of an inherent challenge for verification planning. The worst case scenario for verification planning is when there is an undefined design component or design feature which has no documented description and requires a significant amount of research and effort on the part of the verification engineer to develop a meaningful verification plan. Other challenges to be aware of include the likelihood of frequent design change requests and late design changes which can often invalidate a large portion of work already done in the verification planning process, resulting in lots of re-work, iterations, and sometimes a complete overhaul of the entire plan and verification effort altogether.

Another major challenge to effective verification planning is related to the technology of the tool set available for planning and documentation. The primary objective for the verification planning tool set is to provide organization and structure. There are industry tools available for verification planning purposes. However, internal company tool sets including custom scripts can also be developed to meet the required objective for providing structure and organization. A limited number of built for purpose verification planning tools are available that have been specifically designed for verification plan development. Yet even with the existence of some built for purpose verification planning tools, given the wide variety of possible customized verification planning flows and approaches and methodologies, there is still the potential for these tools not to satisfy every single planning feature desired or needed across all customer use models. Therefore a need to support user specific customizations is vital for purpose built verification planning tools. Even if the capability for customizations is well supported within a verification planning tool, the need for the creation and maintenance of custom scripts outside the planning tool most certainly still exists and can be a significant amount of effort. In order for a purpose built tool to be effective for verification planning, it needs to be a flexible customizable tool that is easy to use, implements all of the basic verification planning methodology fundamentals, and whose source data format supports a wide possible range of scripting. Essentially, a built for purpose verification planning tool must support the concept of creating an executable verification plan.

III. Executable and Living Codified Verification Plan

Verification methodology is a large subject in itself, but one of significant importance and interest across the industry. Companies throughout the industry are invested in creating, maintaining, and continuously improving their verification methodology. In the area of verification planning methodology, the concept of a codified executable verification plan, where the plan takes an active and participatory role in verification, is showing great benefits through the entire verification flow. Based on experience, establishing a flow that begins with and gives adequate time to the creation of a codified executable verification plan implementation is a solution that has addressed many of the barriers and challenges to both the process of effective verification planning and the overall verification flow all the way through closure and completion. Combined with automation, this executable verification plan becomes a single point of source for the entire verification flow.



Specification And Design Requirements
Verification Strategy & Methodology
Verification Plan Template
Reviews And Reporting Plan
Testbench Design
Implementation Planning
Verification Closure Plan
Change Management Process
implementation
Executable Verification Plan and Results
Verification Execution (Simulation, Formal, Etc)
results

Figure 1: Executable Verification Plan Concept

An executable verification plan is a plan that defines the scope of a verification project, describes all of the key features of the design to be verified within the established scope, has a well defined list of testcases/checks/coverage objects identified per feature, and has the necessary design requirements linkage connected to the testcase/check/coverage object level. What makes the plan "executable" is the notion of using it to define and refine your test strategy, *and* the notion of collecting all run time data and results to make the plan dynamic and alive with the execution process. An executable verification plan also needs to have the capability of generating or being able to extract a complete simulation regression list of testcases from the codified feature set and correlating the regression pass/fail results to the objects in the verification plan. Finally, an executable plan needs to have the capability of supporting design requirements linkage from a set of input requirements to the endpoint testcase/check/coverage objects.

Considering the cost of the time needed to implement a well structured and coherent executable verification planning flow, the benefits are typically recovered during the testcase and testbench development because of the structured plan. With the codified structure implemented in the plan, testbench code and testcases can be written more easily as an out-flowing of the planned objects. Additionally with the concept of the executable plan, the final verification result which includes the completion criteria and coverage is already built-in to the verification plan, such that the regression results and coverage are mapped directly to the plan structure and objects.

The following sections will provide details of this claim shown in an example verification planning flow detailed through implementation and closure.

IV. Verification Planning Flow: A Complete Methodology

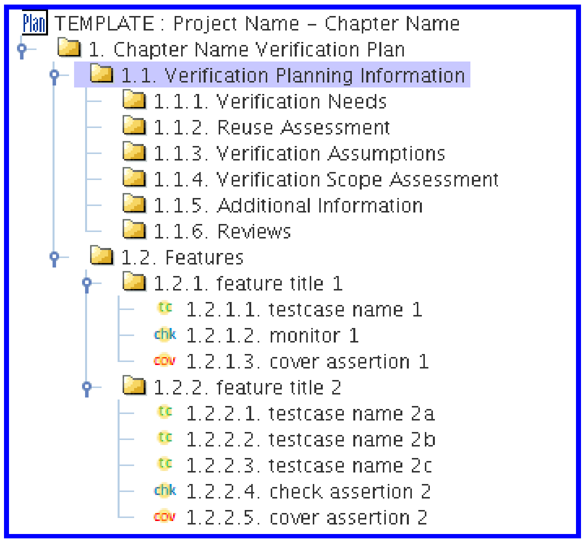
The following example planning flow presented in this paper is one possible implementation of an organized structure which proved to be a good fit for a large scale SoC level verification project within Freescale. It is by no means the only way to achieve effective verification planning, but it does take into consideration the planning process in the context of a larger verification methodology. It has proved to be useful and successful based on experience using it on a large design with a widely distributed verification team.

1. Create a Template
    a. Maintain Consistency
2. Verification Planning
    a. Verification Needs
    b. Reuse Assessment
        i. Design Maturity
        ii. Know Who and What You're Working With
    c. Verification Assumptions
    d. Verification Scope Assessment
        i. SoC vs. Block Level
        ii. System vs. Peripheral vs. Integration
        iii. Directed vs. Randomized vs. Formal
3. Implementation Planning
    a. Requirements Driven
    b. From Specification: Feature Driven
    c. Prioritization of Features
    d. Organization of Features/Requirements
4. Closure Planning
    a. Coverage Correlated to Verification Plans
    b. Completion Criteria
5. Reviews / Reporting

Create a Template

Creating a solid verification plan template is crucial in order to establish and maintain consistency across the individual verification plans feeding into a larger verification guide. The importance of having a verification plan template is to provide the multiple individual verification plan developers on a project with a prescriptive development approach to follow, guiding each individual through a consistent plan creation process.

The template for the verification plan should be created directly within the target documentation/planning tool that is to be used in the project for all the verification planning, i.e. if the verification plan tool to be used is a spreadsheet, the verification plan template should be created as a spreadsheet with the basic template structure. The template should contain well defined sections with all the necessary subsections, and clear description within each section regarding the expected content and style required for that section. The following picture shows an example template structure with the basic plan sections ready to be modified by the individual verification engineers:



```
TEMPLATE : Project Name - Chapter Name
  1. Chapter Name Verification Plan
    1.1. Verification Planning Information
      1.1.1. Verification Needs
      1.1.2. Reuse Assessment
      1.1.3. Verification Assumptions
      1.1.4. Verification Scope Assessment
      1.1.5. Additional Information
      1.1.6. Reviews
    1.2. Features
      1.2.1. feature title 1
        1.2.1.1. testcase name 1
        1.2.1.2. monitor 1
        1.2.1.3. cover assertion 1
      1.2.2. feature title 2
        1.2.2.1. testcase name 2a
        1.2.2.2. testcase name 2b
        1.2.2.3. testcase name 2c
        1.2.2.4. check assertion 2
        1.2.2.5. cover assertion 2
```

Figure 2: Example Verification Plan Template

The verification plan template should also include detailed explanation descriptions (not shown in figure 2) in the information field of each section folder which describes clearly what each verification engineer should include for that section. For example, the section for "Verification Needs" has an information field which states:

> *"You shall create a documentation object in this Verification Needs folder for each need associated with your verification project. The needs you indicate shall include the following required objects: design document locations, special testbench drivers/ monitors, etc..."*

Verification Planning
The next step in the planning process is to edit the template and enter specific information into all of the planning sections for the target verification project. In this example, the planning sections are:

- Verification Needs
- Reuse Assessment
- Verification Assumptions
- Verification Scope Assessment

In the section for Verification Needs, the verification engineer needs to investigate and document all of the verification needs for their target verification project. This should include identifying needs such as:

- All of the design documents that will be used as inputs for the target Device Under Verification (DUV)
- What is the requirements tool to be used for linkage, and how will the requirements be identified for the target DUV?
- Whether there will be behavioral models or functional models needed as part of a system in the target verification project?
- What are all of the design blocks which will be needed to interact in the system that can't be modeled?
- What are all the protocols used in the target DUV?
- What are the testbench components that are necessary for the target verification project?
- Purchase vs. make: needs analysis

Reuse Assessment
Each project is a unique and complex amalgam of design and testbench components. While not directly related to the concept of a living codified executable verification plan, the reuse assessment is a crucial part of an effective and complete verification planning methodology in order to identify up front what is the maturity of existing design and verification components that will be used, and if the components identified during the Verification Needs analysis are not available and need to be developed. The reuse assessment of

verification collateral also provides an opportunity to review the verification and design material up front in the project in order to understand what exists and what is needed going forward.

It is also vital to identify who are the designers that the verification team will be working with on the project. This should be done during the reuse assessment in order to establish the expectations of engagement and establish what the verification team can expect to receive regarding the level of reuse. Another area of potential reuse to assess is what other verification resources are available during various stages of the project. Often times, projects utilize some level of reuse both in the design and the testbench, but without a careful and proper assessment of the realistic level of reuse, too often the assumption is that "reuse" means 100% reuse and the project ends up under-resourced and under-scoped.

The importance of the reuse assessment is to identify for every piece of design code and verification code, what exists and to what extent is the code reused, modified, or new.

Verification Assumptions
The next step in planning should be to document and review any assumptions being made by the verification team for the project. All of the assumptions should be reviewed by the verification team with the designers such that there are no surprises late in the project and such that the team has a chance to challenge any bad assumptions. For this review to have the best effect face to face meetings are crucial and it is common to have several reviews to cover all the assumptions before alignment is reached.

Some examples of verification assumptions could be similar to the following:

- "Feature X cannot be verified in digital simulations; requires Analog Mixed Signal(AMS) simulations to verify, not verified under this verification plan."
- "Feature X should be checked by Block Z instead of Block W - not verified in this plan."
- "System level signal Y cannot be adequately modeled in this standalone block level testbench, corresponding feature X must be verified during SoC verification."
- "Use of the standard protocol driver does not implement the custom extension to the protocol mentioned in the design spec, but the project will not implement the custom extension. Thus the existing standard driver is adequate as-is."

Assumptions all too often are left unspoken and undocumented, and can lead to design features falling through the cracks. All assumptions should be very closely tracked, reviewed, and scrutinized. One method of tracking the verification assumptions resolution is to document all the findings and conclusions arrived during the reviews in this same "Verification Assumptions" section of the plan.

Verification Scope Assessment
The purpose of the Verification Scope Assessment is to identify the target verification approach to be used for the verification project. A limiting factor may be identified during the reuse assessment in the case a legacy verification flow or methodology was previously used and the cost of rewriting the verification testbench for a newer and better approach might not be feasible. Another limiting factor might be the limited skill set of the verification engineers assigned to the project. Apart from these limitations, the Verification Scope Assessment

should identify the best approach to follow for the target verification project. Assessing the following criteria can help identify what is the scope of the verification approach for the project:

- Identify your group's documented and prescribed verification methodology standard to be used for the project.
- What is the availability and quality of the design requirements and specification documentation?
- Is the target DUV SoC level, Sub-System level, or Standalone block level?
- Is the verification project primarily an integration focus or a full feature focus?
- What is the level of analog content of the DUV and what type of analog views will be used?
- What metrics will be used to measure coverage and closure - assertions, self-checking, visual inspection?
- Are there any checklists which are required by the methodology standard or project procedure that affect verification decisions?

This assessment should help identify the verification scope, whether the target verification project should use a directed self-checking stimulus approach, or use a transaction level constrained random stimulus approach, or if a formal verification approach is the best solution. Additionally, this assessment should help identify what amount of check and coverage assertions should be necessary in the verification project to satisfy the coverage and closure criteria. Also, the verification methodology should be able to identify what types of coverage should be required for the verification project as a result of the verification scope assessment - i.e. HDL Toggle coverage only, full HDL code coverage, functional coverage, testbench coverage, etc...

It is worth noting that there are a wide range of valid verification approaches that should be considered in determining the appropriate solution for each verification project. The purpose of the scope assessment is to determine how to best use the limited amount of verification resources available in the most effective and efficient manner to attain the highest overall value of quality, time, and effort. The identified solution should always be in line with the established verification methodology guidelines. If guidelines or checklists exist in the verification methodology, the scope decisions should be driven by these guidelines.

Implementation Planning
Planning the implementation of the verification project is really the essence of successful verification planning. The implementation planning should be driven from the following sources:

- Requirements Driven - for requirements linkage
- Feature Driven - from design specification documentation
- Priority Driven - for projects with a design priority implementation flow

The organization of the features to be verified in the verification plan should be well ordered to maintain coherency. There are many different valid verification approaches that might lend themselves towards different organizational structures within the verification plan implementation section. The following example verification plan (see figure 3) shows a picture of a directed self-checking stimulus flow for a serial interface block to be integrated in an SoC. The structure pictured in this example is just one possible method to organize the plan according to a feature based verification flow:

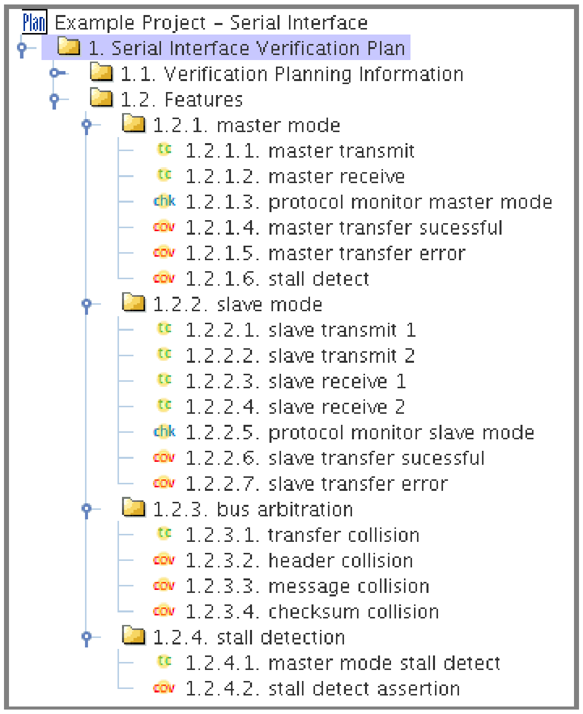

Plan Example Project - Serial Interface
1. Serial Interface Verification Plan
1.1. Verification Planning Information
1.2. Features
1.2.1. master mode
tc 1.2.1.1. master transmit
tc 1.2.1.2. master receive
chk 1.2.1.3. protocol monitor master mode
cov 1.2.1.4. master transfer sucessful
cov 1.2.1.5. master transfer error
cov 1.2.1.6. stall detect
1.2.2. slave mode
tc 1.2.2.1. slave transmit 1
tc 1.2.2.2. slave transmit 2
tc 1.2.2.3. slave receive 1
tc 1.2.2.4. slave receive 2
chk 1.2.2.5. protocol monitor slave mode
cov 1.2.2.6. slave transfer sucessful
cov 1.2.2.7. slave transfer error
1.2.3. bus arbitration
tc 1.2.3.1. transfer collision
cov 1.2.3.2. header collision
cov 1.2.3.3. message collision
cov 1.2.3.4. checksum collision
1.2.4. stall detection
tc 1.2.4.1. master mode stall detect
cov 1.2.4.2. stall detect assertion

Figure 3: Example Feature-Based Organization

The benefit of this level of organization during verification planning is that the actual implementation of the testbench and stimulus should be a direct out-flowing of the structure identified in planning. The actual work of writing stimulus and testbench components from a well ordered organized plan is substantially more complete than an ad hoc approach and the structure of the stimulus in the executable verification plan means that the regression list of stimulus is already created and maintained in a single source, and the linkage to requirements, coverage, and pass/fail results is also already created and maintained in a single source. This ability to roll up the results for closure in a single executable verification plan brings the process of planning to closure into a full circle.

Utilizing Tool Customizations and Creating Custom Scripts

The example Freescale SoC verification project discussed in this paper required a number of verification planning tool customizations to support the identified verification planning needs. Two examples of customizations Freescale needed were:

- Custom field for requirement tags linkage
- Custom field for codified target simulation configurations, i.e. rtl/gls/ams

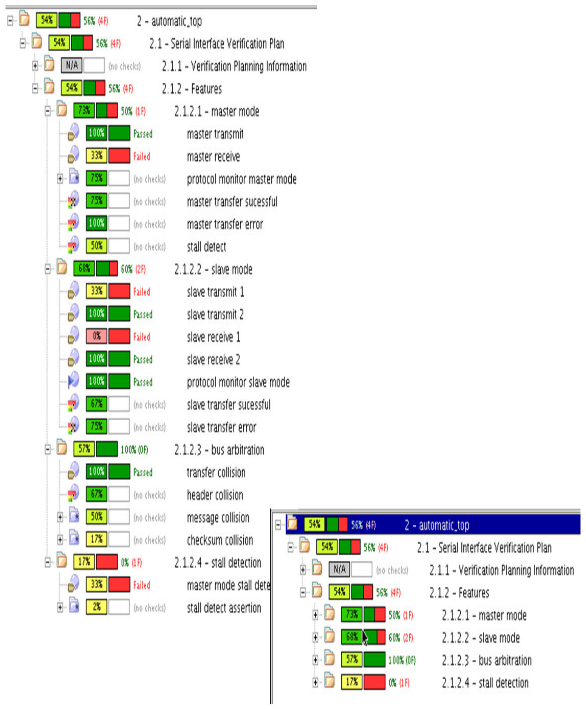
Regarding the requirement tags linkage, Freescale added a user specified field in the verification plan which allows the verification engineers to insert requirements tags directly connected to the testcase and coverage objects. These requirements tags and corresponding testcase/coverage objects are then linked to the design requirements using a separate requirements traceability linkage tool.

Regarding the customization for supporting target simulation configurations, another user specified field was added to the verification plan allowing verification engineers to specify the target simulation session per testcase object. For example, one testcase might be targeted to only run in rtl simulations, and another testcase might be targeted for gls and ams simulations. In the example verification project, this customization is important because the executable verification plan is used as the single source for the regression testcase list, and therefore the verification plan needs the capability to add the target simulation configuration attribute to each testcase.

Creation of a custom script was also required in conjunction with the target configuration customization mentioned in this example. The executable verification plan needs the ability to generate or have extracted the full simulation regression list of testcases with the target configuration parameters codified in the list. However, the verification planning tool used in this example does not have the built in feature to natively generate the necessary regression list in the target format needed. Therefore, the need to create a custom script to automate this extraction process directly from the verification plan source was critical to the success of using the verification plan in the project. The example custom script is written in PERL and it parses the verification plan source to locate valid testcases and extract all of the relevant parameters codified in the plan for the regression list. A crucial element of this process is that the verification plan tool must provide a format of the data that can be parsed in a reliable and repeatable manner.

Closure - Metrics

In planning, it is important to determine the signoff and completion criteria according to the verification methodology. This should correlate with the assertions identified during the implementation planning, and should also be consistent with the verification approach identified during the scope assessment. The benefit of creating and using an executable verification plan is that it serves as a single point of source throughout the entire verification flow that allows the results to be correlated according to the plan and in conjunction with the plan of record. Closure should include the ability to link all of the coverage and pass/fail results to the feature set created during planning, and support any requirements linkage identified during planning. The picture shown in figure 4 gives a visual example of how the pass/fail simulation results and the coverage results can be linked directly back to the example executable verification plan structure seen previously in figure 3:



```
54%   56% (4F)      2 - automatic_top
  54%   56% (4F)      2.1 - Serial Interface Verification Plan
    N/A   (no checks)   2.1.1 - Verification Planning Information
    54%   56% (4F)      2.1.2 - Features
      73%   50% (1F)      2.1.2.1 - master mode
        100%  Passed        master transmit
        33%   Failed        master receive
        75%   (no checks)   protocol monitor master mode
        75%   (no checks)   master transfer sucessful
        100%  (no checks)   master transfer error
        50%   (no checks)   stall detect
      68%   60% (2F)      2.1.2.2 - slave mode
        33%   Failed        slave transmit 1
        100%  Passed        slave transmit 2
        0%    Failed        slave receive 1
        100%  Passed        slave receive 2
        100%  Passed        protocol monitor slave mode
        67%   (no checks)   slave transfer sucessful
        75%   (no checks)   slave transfer error
      57%   100% (0F)     2.1.2.3 - bus arbitration
        100%  Passed        transfer collision
        67%   (no checks)   header collision
        50%   (no checks)   message collision
        17%   (no checks)   checksum collision
      17%   0% (1F)       2.1.2.4 - stall detection
        33%   Failed        master mode stall dete
        2%    (no checks)   stall detect assertion
```

```
54%   56% (4F)      2 - automatic_top
  54%   56% (4F)      2.1 - Serial Interface Verification Plan
    N/A   (no checks)   2.1.1 - Verification Planning Information
    54%   56% (4F)      2.1.2 - Features
      73%   50% (1F)      2.1.2.1 - master mode
      68%   60% (2F)      2.1.2.2 - slave mode
      57%   100% (0F)     2.1.2.3 - bus arbitration
      17%   0% (1F)       2.1.2.4 - stall detection
```

Figure 4: Simulation Results Metrics Report

Reviews / Reporting

Planning for proper reviews and reporting should identify the reporting methods to be used and the stages where different reviews need to be held during the verification flow. The topic of reviews is indeed a large subject in itself, but it is worth noting the extreme importance of holding proper verification reviews with the proper audience at various prescribed stages of planning, implementation, debug, and closure. Representatives from the design team, test team, systems team, and verification peers should all be included to participate in the verification reviews. A major benefit of an executable verification plan is that the review records can be linked directly to the plan objects and easily accessible in the case of audits and quality reviews.

V. Summary/Conclusion

Verification planning is not simply a task that is done once and forgotten at the start of a verification project to satisfy some management check box. Verification planning is a living, breathing and executable methodology for saving time and valuable resources throughout the life of a project. Like most methodology changes however, there is always a cost. In the case of verification planning, the cost is in adherence and discipline, but with a tremendous set of benefits. With an organized verification planning methodology and management solution, the typical manual and arduous reporting process comes for free, the analysis of what was tested and what was not is fully automatic, and the implementation details to achieve closure takes virtually no work as information is alive and independent of the testbench. In our experience, a meaningful verification planning process may add 10% to 20% to the verification frontend. But in return the verification planning investment can save 10% to 40% in the overall verification schedule, plus recover the planning costs, plus provide reduced direct effort around verification implementation and closure by at least 20% (see figure 5)

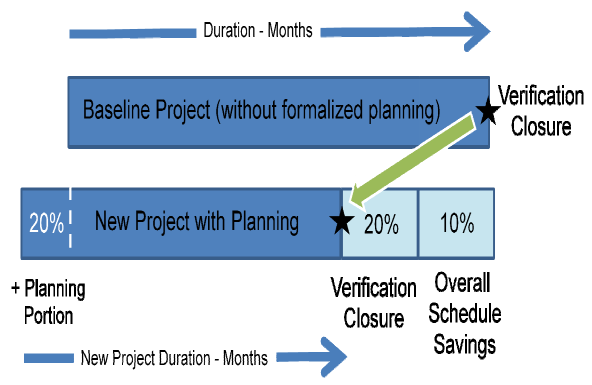


Figure 5: Value of Verification Planning

When verification planning is given proper focus and priority, the overall verification project flow has coherency using a single point of source for development of the verification collateral by strongly correlating simulation results to feature closure. Implementing an executable codified and feature based verification plan is a highly beneficial enhancement to any verification methodology.