



Verification Planning and Metrics to ensure efficient program execution

**DVClub: RTP, North Carolina
Jan 17th, 2007**

Joe Rash

Joe is currently working for CebaTech Inc. where he is responsible for product management and business development of their IP product lines.

Prior to joining CebaTech, Joe held a number of management and leadership positions at AMCC, Nortel Networks, and IBM. As the Director of engineering at AMCC, Joe was responsible for company wide verification methodology, instituted and sponsored a company wide verification users group, and had development responsibility for a number of complex framer and network processor ASICs.

Thanks and disclaimer...

“Thank You” to the folks that participated in the DVClub advisor’s lunch and provided a lot of good ideas for this presentation.

This presentation is meant as a guidepost for improving the verification planning process. There are numerous approaches that teams go through during the plan creation phase, and an even greater number of formats for the plan itself. This set of “helpful reminders” will hopefully benefit you when you tackle your next verification plan for your company.

Get off my back hot button

When you see this  pay close attention. This is a manager hot button. Anything you can do to mitigate or address this item will help get your manager off your back!



Verification Planning

What's all this about planning?

The U.S Military. Organized, efficient, and some of the brightest and most courageous leaders on the planet have an old axiom that reads:

“It's better to have a poor plan well executed than a great plan poorly executed”

But is it true?

Look no further than Iraq to see an example of a “poor plan” that has been “well executed” to know that you need a great plan too!

No doubt about it, people come first and a great team can make miracles happen!

The most successful teams have great people, and a well thought out and documented verification plan.

Common Issue

All too often ASIC teams make the mistake of drafting a half baked plan and launching into the environment build process driving feverishly for the first evidence of “simulation life” in a new design. Engineers generally do not like planning...

The result:

- Poor predictability 
- Environment churn at inappropriate times in the project
- Inadequate control or observability
- Missing coverage

Remember: **Plan** → **Build** → **Test**

What is the objective of the plan?

Now that we have established the importance of planning. What is the objective of the plan itself?

The plan(s) should comprehensively convey:

- ✓ **Detailed functions covered/tested**
- ✓ **How exhaustive testing of the DUT will be accomplished**
- ✓ **Completion Criteria (which is often prioritized)**

Items frequently neglected in the test plan 

- ✓ **Gate level testing**
- ✓ **Testing of the DFT functionality**

What should be considered before drafting the first plan?

Explore the feature set of the DUT (begin with market requirements)

How does the DUT function in a system?

What major functions does the DUT perform?

What are the major interfaces and are they standard or proprietary?

Think about different ways to isolate and control stimulus

Think about how you might check the function (modalities)

How many environments might I need to get the job done?


What are my reuse opportunities?

- ✓ **From previous projects**
- ✓ **From block to top level environments**
- ✓ **Others (test case, assertion, etc.)**

The plan and planning process

The plan itself can be thought of as a set of plans:

1. Comprehensive Coverage Plan

- ✓ Cover items
- ✓ Assertions
- ✓ Formal checks
- ✓ Dynamic embedded checks
- ✓ Directed testing
- ✓ Manual checks 

The better you define the “coverage mechanism” the more likely you are to have “right sized” your environment plan...

2. Environment Plan

- ✓ Block Environments, Checkers, Checker reuse
- ✓ Full Chip Environments (including DFT and Gate sim)
- ✓ Transactors, BFM, and 3rd Party Model Usage
- ✓ Co-simulation, software reuse, automation, etc.
- ✓ How will the environment itself be tested

Manual checks should be minimized but if needed should have gates in the automated regression runs to prevent advancement without checking

The plan and planning process continued

3. User Guide Run Time Plan

- ✓ How to build, make, run
- ✓ Regression management
- ✓ Ranking, Pruning, Refining the test suite
- ✓ Environment Profiling to improve run time
- ✓ Extended Random testing

4. Completion Criteria and Plan Metrics

- ✓ Interim targets against the environment build process
- ✓ % written, % run, % covered, ... as a measure of your comprehensive coverage plan items
- ✓ Code coverage – this is a secondary measure of your plan quality!
- ✓ Extended random testing completion (time based, bug rate based, other)
- ✓ Reviews completed (Plan, Test cases, cover items, etc.)

GOMB

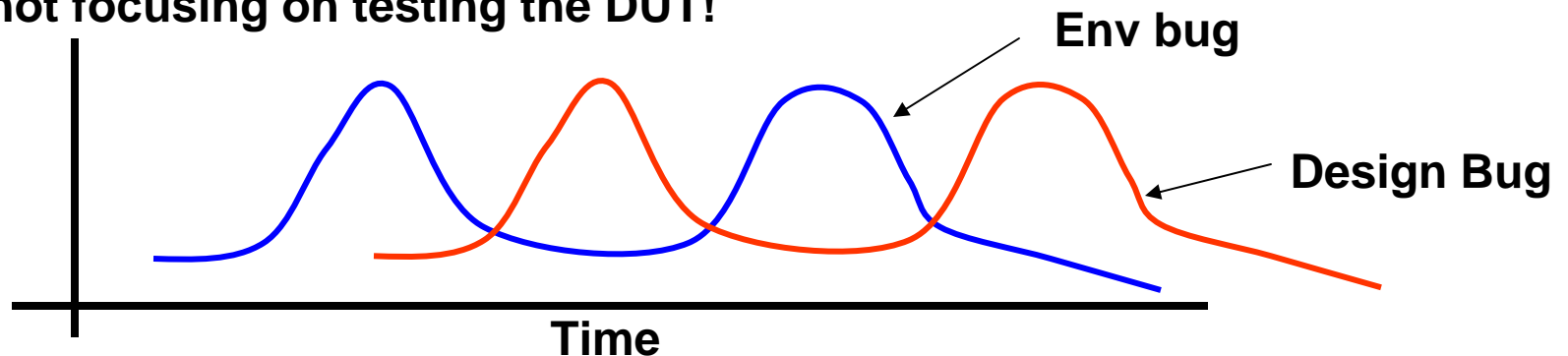
Additional metrics

In addition to tracking design bugs and the usual verification progress metrics, I have found it useful to track bugs and issues in the environment or test cases.

A bug filed against the environment indicating “function missing” in the environment (i.e. no way to stimulate, or no way to check) is actually a good indication of a breakdown in the planning phase!

The trends are usually more useful than the discrete numbers!

We always saw an interesting correlation between environment bugs and design bugs. If the team is spending their energy fixing the environment, they are not focusing on testing the DUT!



Summary

The planning process should be iterative and top down. Comprehensive coverage planning with thought about “how” to most effectively cover the function will guide the environment planning.

- ❑ Eliminates unnecessary complexity in the environments**
- ❑ Ensures the necessary features in the environments are included up front**
- ❑ Improves run time by using the most effective means to target a function**
- ❑ Helps to identify tools, IP, and training early in the process**

The plan itself may be a set of plans. It is dynamic and ever changing, not a static document.

Capturing completion criteria, including interim environment build milestones, will help provide a progress report during the verification phase. Exit criteria as part of the initial planning will provide a guidepost for completion when you enter the “are we done” stage at the end.

Q & A