In today's short post what I'll try to do is share with you some of the recent trends and ideas that deal with coverage. I won't go into much technical detail today in order not to wear you (and myself) out, but really - if I want to be more catholic than the pope (the pope of verification that is), each of the terms I'm going to mention probably deserves its own article.

First things first - here's something I'm a big advocate of - The Plan Driven Verification. For me, the Plan Driven Verification approach (let's just call it PDV ok?) is not just a new trend, but is mostly valued for what it's not - Coverage Driven Verification. I'll explain what I mean.

The straight forward CDV, or as I like to call it - Raw CDV - was a real breakthrough in the perception of functional verification and is indeed a powerful engine. No need to babble about it too much. But, it lacked a few things that could have made it an excellent methodology, not just good. I believe some of the inherent problems of Raw CDV are addressed by the PDV. Let's talk about them shall we?

Raw coverage information is a number, or a set of numbers that tell you exactly HOW MUCH is covered, but gives you little or no information about WHAT is covered. Now, as most verification plans originate from some sort of Spec, which is a feature-oriented document (I would even say human-oriented), analyzing a raw coverage report without having it communicate with the feature it originated from is problematic and ineffective. I've seen a lot of cases where people talk enthusiastically about their block coverage hitting 90%, but when asked what's hiding behind the 10% coverage hole, they didn't have a clue PDV solves that.

But not only that, PDV allows you to separate **WHAT** you want to verify from **HOW** you're going to verify it. What if you only want to use a set of 10 plain old directed tests to cover a certain feature? What if a specific feature is covered by an assertion? PDV gives you the flexibility to do that.

As a matter of fact, I would say that Plan Driven Verification kind of makes you enjoy both

worlds -
your still can use Coverage Driven Verification as a basis, and at the same time your progress metric is no longer a vague number but rather - a meaningful report that has all the necessary details.

A word of caution though, Plan Driven Verification requires additional effort at the planning phase as every single coverage item is required to be linked to the appropriate feature in the top level verification plan.

PDV might also help verification guys be less exposed to what I call Coverage Attacks. The problem is that sampling coverage on a weekly basis might not always show linear progression. The numbers might go down occasionally. Now managers don't like to see this, and the result is that too much focus is put on showing politically correct coverage results, while less focus is put on understanding **WHAT** has been covered and what has not.
Just something to think about..

And 2 more tips about coverage in general. First, if you collect coverage reports on a periodic basis - coverage progress may be extrapolated to estimate convergence to coverage goals. I know people who have made this a real science.

Second, for the purpose of tracking the DUT's readiness for tapeout, coverage data is typically collected from the passing tests only. How else, right? but believe it or not, we can get additional (and important) information by looking at the coverage results of the entire regression, including failing simulations. This is how you can track the **ENV** readiness (generation matureness).

That's pretty much it for today. Like I said before - each of the terms mentioned here probably deserves it own discussion - CDV, PDV, coverage extrapolation, etc. So if any of you guys feels like sharing your view on these subjects - write down your thoughts and send them over. I'll host you as a guess blogger here on Think Verification.

And to conclude today's post with a smile - some good news from Specman! Does long lines of type casting sound familiar?

var x:= struct_a.as_a(TYPE1 struct_a_s).struct_b(as_a TYPE_X struct_b_s).struct_c.TYPE3.field_a

Well, rumor has is that all this is going to be history with Specman's auto type casting in the next release. Sweet!