Is Assertion-Based Verification (ABV) becoming mainstream? This question popped up today at Mentor's ABV seminar . Assertions in general and ABV in particular make another approach that you can use to verify your design. Usually ABV alone is not sufficient, and is used alongside other verification approaches such as Coverage-Driven Verification , Directed Testing, Formal Verification, and Code Coverage. What's good about ABV though, is that it's the fastest tool around in terms of failure-to-root-cause time (also in your testbench !)

In other words, if you have a failing assertion during simulation – probably the bug it not too far away. What's more, assertions catch bugs immediately as they occur, rather than thousands of cycles later when the bug has (hopefully) propagated out of the chip. However, there's no such thing as a free lunch. There is a price you have to pay for being able to pinpoint bugs as they occur. Obviously somebody has to write the assertions and then make sure they are capable of catching what they should catch. But then one might ask – Ok, so if I start using assertions will I be able to do less "traditional verification"? Can ABV safely replace some of the verification techniques that I'm currently using today or does all this translate into extra work?

Two things about ABV you might want to consider. First, while verifiers are responsible for both defining the test plan and implementing the testbench, in ABV things might be a bit different. Properties of internal logic might be too difficult for verifiers to implement using assertions. These properties are better left for designers to code. And let me tell you something – designers don't always like to play along. Second, regarding reusability – a set of assertions is probably the most reusable element of your testbench that you can take "as is" from your block level verification to higher scopes of verification. If you're an IP vendor, assertions are also a great way to embed testbench elements in your product and gain valuable debug information on the field.

And lastly, if debugging accounts for 70% of design effort, then adopting a tool that significantly shortens the failure-to-root-cause time should be favorably considered by VLSI managers (once they've realized that verification is no magic ). Does this mean that ABV is becoming mainstream? You be the judge.