Things are changing. The EDA industry is changing, and the verification world is changing (check out Janick Bergeron's inspiring talk at SNUG San Jose for a glimpse of the future of verification). One of the major challenges we're already facing today is an increasing number of IP blocks in a single system. Designs are getting bigger and bigger, and the focus is shifting towards system level integration rather than "design creation".

If you look at some design teams today, don't be surprised if you meet engineers who don't understand what their chip does, and only know they need to integrate it. This raises an obvious question as to what level of verification quality engineers could possibly provide if they know nothing about the internals of their chips? The industry is beginning to address this question. Perhaps a significant improvement in the area of "correct-by-construction" techniques, along with an industry standard of "IP-level" verification quality stamp would do the trick. Nevertheless, none of these solutions could solve the problem completely unless you go higher in abstraction level. In the RTL world flaws and coverage holes will always remain if you let one team design and verify your cores and another team integrate and verify the system, unless you allow for redundancy and transfer of knowledge and expertise. Problem is, who can afford this? Wouldn't it be better if you could buy your IPs off the shelf and never worry about bugs that might be lurking inside?

But that is the smaller problem. The bigger problem is that an off-the-shelf design IP often requires special configuration and tweaking before it can function smoothly in its target system. And guess what happens if both IP design and system integration are done at the RTL level? That's right. Simulation, verification and modifications are done at the RTL level as well! Apart from presenting heavy requirements on computing resources, this also distracts chip integrators from their real focus. As chips are becoming more and more gigantic, chip designers should be focusing on system level simulation. This means that IP cores should be modeled at a higher level than RTL. At a high level of abstraction there won't be any "black holes" in your design that you don't understand because implementation would become irrelevant. Let the tools deal with implementation while you enjoy an RTL-free world. Your IP blocks would be highly configurable and integration-friendly thanks to a rich set of API functions and settings - all provided by your IP creator. That is perhaps the most significant trend verifiers should be expecting in the future. The EDA industry has already taken a few concrete steps towards the visionary application/integration driven world, one of them is the recent release of the UVM methodology , aiming to make all other verification methodologies a fading memory.

Now what?

But until the dream comes true, let's look at what we can do today to solve at least some of the problems in today's integration-focused verification efforts. We're still talking RTL, no big news there, just a few relatively quick and easy optimizations you can make in your top level verification environment and verification strategy. Nevertheless, these are powerful field-proven techniques that have helped to achieve a significantly improved performance and faster results in real-life projects.

Simulate only what's necessary and nothing else

At the top level verification phase, different tests verify different parts of your design. Unless you have a good reason why all your IP cores should be simulated and verified in all of your test cases, you can safely remove any part of the design that's not being verified in a given simulation. Sometime this is as easy as internally wrapping all functional blocks with "ifdef" macros. Each testcase would then activate only the necessary blocks for simulation. For example, if your system consists of 10 IP cores, and in a certain simulation only 2 of them are being activated and verified, remove the other 8 in that simulation. The less RTL code that's being simulated, the faster your simulation will run.

Build your environment dynamically

And the same goes for verification IP – remove any verification IP that's not absolutely required in a given simulation. For that you'll have to use a dynamic building approach in your verification environments rather than a "everything-is-instantiated-all-the time" approach. In terms of top level verification strategy what we're doing here is applying the "verification flows" technique in which it is assumed that top level functionality can be separated into several independent flows whose "sum" is equal to running all flows at the same time, at least in terms of coverage. This superposition technique might not be valid in all cases, but if it is in your case, you got yourself a wonderful way to speed up simulation performance and you pay nothing for it.

Use loopbacks creatively.

The idea behind reusing IP cores is that you focus on system level integration while assuming that your IP cores are pre-verified. This sometimes means that there is absolutely no need to place verification components on each and every port or interface of your design. Instead, you can use simple loopback setups to route traffic from one interface directly to another and let the design itself play the roles of monitor and driver. For example, let's say your system has a serial 10GBE interface at one end, and a 4-lane XAUI interface at the other end, and your current verification IP can talk only with the serial interface. You can connect the TX port of the XAUI interface directly back to its RX port with no verification stuff going on in the middle. Assuming your only goal is to verify that the XAUI block is well integrated in the system, the loopback technique may be all you'll ever need. People sometimes underestimate the effort of developing verification IPs for standard interfaces. Even if you're reusing verification components from previous projects or commercial ones, maintaining them and learning how to use them will require a significant effort. If you have a silicon-proven 100%-verified IP core in your system that you can use as an "indirect driver" this could be a great alternative to a fully blown verification component.

Look under the street lamp

It's good to keep in mind that many standard protocols provide testing features that can be recruited to your verification needs. Moreover, IP creators often throw in a bunch of verification-friendly engineering features such as internal error injection machines, or random payload generators that can be easily turned in to verification agents serving your needs.

Use assertions

What's cool about assertions is that IP creators build them, but integrators use them. Assertions are little spies in your designs that are capable of catching functional bugs the very moment they occur. Encourage your IP creators to place assertions wherever possible and make sure you enable them in your system level simulation. Read here for more about assertion based verification.

One final remark on the benefits of using a compact verification environments and eliminating unnecessary components where loopbacks or design-for-verification features can be employed. The less verification components you have the less pain you have during integration of verification conponents. This means that system level verification can begin earlier and time wasted on debugging the environment can be spent debugging the RTL. And the more straight-forward your verification environment is the easier it is for designers with no verification expertise to feel comfortable with the verification environment and take on verification and debugging activities with minimum support from verifiers.