UX Design & EDA – Enable Collaboration on Functional Coverage using Opens Source Software

Coverage becomes more and more important in the verification process as resources are costly and besides bugs it is the only real metric in the validation process. And there are many technologies to take advantage of the coverage events. Using analytics and AI to improve the testbench parameters based on coverage information is one example.

How do we make sure we have the right events defined? Who is creating the cover points? When in the process is are coverage events being created and even more important - when are they getting looked at? And how can they be hit?

How does a verification engineer know which coverage hole is important and how to go after hitting it? Are all important coverage events defined?

Automated tools to create toggle, line, state machine coverage or even property synthesis by analyzing simulation results can create a lot of events but give very little guidance on what is important and does not cover the complex, deeper events and not the ones we don't hit. Both designer and verification engineer are spending a lot of time analyzing unimportant or even irrelevant coverage holes.

The Logic designer is busy with creating logic designer so that the verification process can start, later he is busy fixing bugs so that the verification engineer runs cleanly, soon timing and integration requires the attention of the logic designer. And where does coverage fit in here?

The obvious answer is that cover point creation and analysis should go side by side throughout the whole implementation process so that closing coverage does not become a afterthought with late hole discoveries leading to last minute bug findings jeopardizing the planned tape-out date.

In addition, a lot of simulation compute resources are being wasted because we don't know whether we run the right things so running more will find more bugs.

What can be done to improve the situation? How can the process be supported without adding more burden to the development team? This is where user experience design become so important and critical for the success. Instead of our classic requirement creation that will lead to a completely overloaded process which ends up not being used.

Starting with user experience research we conducted more than 30 user interviews of verification engineers and logic designers to identify pain points. We use enterprise design thinking to create as-is-scenarios, user needs and ideas. We did user playbacks to get feedbacks, we created multiple design prototypes.

We very soon found out that the collaboration between the design and the verification engineer on coverage definition, creation, prioritization and analysis is the key to significantly improve the process.

We created nonfunctional design prototypes (yes to demonstrate the UX not the function) , did playbacks with the users on every step. And created functional prototypes that demonstrating the technical feasibility.

We will present the use case for an interactive collaboration system that enables the design and verification engineer to collaborate from the feature creation to coverage closure, which is truly driven by user experience design. We will explain the advantage of using user experience design in developing eda tools and the process we went through to create these user experiences using the open source Carbon Design System to collaborate on functional coverage with demonstrating an example based on Cocotb, an open opensource python based verification framework and a Python based UVM Coverage library (Pyvsc) to create coverage points, bins and cross product coverage as an example.

At the end of the tutorial, the attendee will understand why and how user experience design and chip development can complement each other leading to much better results in the chip development process.