Introducing Crossfire Diff

A Fractal Whitepaper

Introduction

It's not often that a single button in a software tool can justify an article all to itself. The new Crossfire diff option, or Cdiff for short, is one of them. Who better to ask for details than Johan Peeters, the inimitable CTO and co-founder of Fractal Technology?

Q: **What is "Cdiff" about for a Crossfire user?**

Johan: 'The Cdiff feature came about from discussions with users. Crossfire has a pretty active and explorative user-community; they typically grasp the technology we're providing and then mold it to do just what is needed for their typical application and design flow. Which is what you'd expect from the design teams designing the most advanced SoC's to market today. Anyhow, what turned out to be a frequent use-case was inspecting a new set of models to find any differences with a previous generation. This is something Crossfire could already be made to do either in the GUI or through our Setup API, but having Cdiff as a button in the GUI turned out to be quite useful.

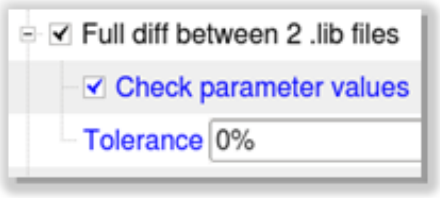

Figure 1, Cdiff switch in GUI

Q: Can you give some real-life examples?

Johan: Imagine your IP supplier comes with new .lib files in which some power arcs are supposedly improved. The first thing you ask yourself on receiving the shipment is: "Did they actually deliver these changes?" And after that: "Did nothing else get messed up on the side?" Crossfire can of course tell you directly if the new library still makes sense from a quality perspective, but for a first inspection you just want to see the changes: is this what I expect to receive, is it sufficient and without side-effects? Cdiff will tell you exactly which arcs have changed and what other modifications occur in the new model as a bonus.

Q: But I could see that using tools like gvimdiff, couldn't I?

Johan: Not really. Remember that this is Crossfire technology we're talking about. So if we'd be comparing 2 .lib files any reformatting of, say, the indent-levels would completely mess up a plain gvimdiff. Tweak gvimdiff to deal with blanks, and you might be faced with differences in the listing of pins or arcs. Cdiff on the other hand would not report any differences on indentation or pin-orders at all, but only flag for example an extra Reset-to-QN arc on one of the flops. We then do make use of the functionality of gvimdiff by using it in our reporting. For this example Crossfire will show the 2 .lib files in gvimdiff, with identical cell order and text formatting where it highlights, only the extra arc in the second .lib file. The messages clearly indicate where to locate it in the original files.

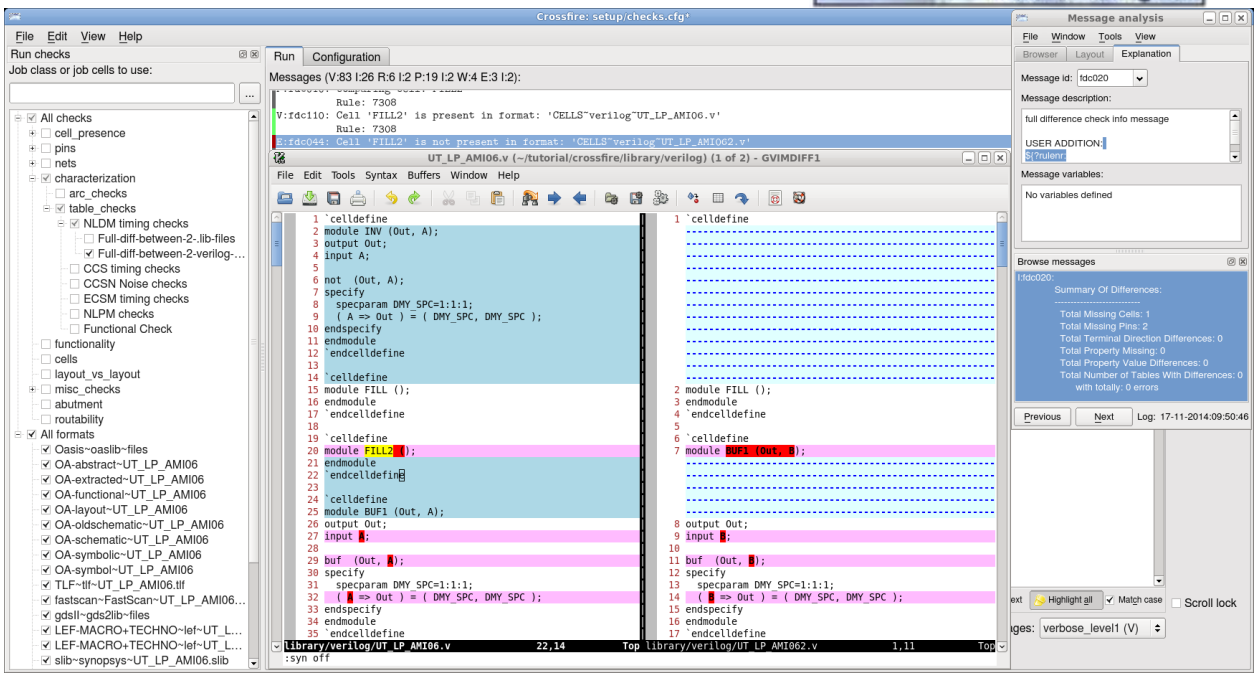

Figure 2, Screenshot of Cdiff output using gvimdiff

Q: So Cdiff is picking up the real issues from the formatting noise?

Johan: That's a good way of looking at it. The amounts of data are enormous these days, our customers characterize literally 100s of different process conditions. If you receive a new release of one of those libraries, the smallest formatting difference between old and new already ensures that you'd never see a real arc-difference when using gvimdiff.

Q: You raise an important point here: why would I care as I already have Crossfire to qualify my new library? I just run the new version through Crossfire again and I know whether it's OK or not?

Johan: Well that's what you will do anyway before taking the new version into production. But you need to see this in the context of a converging product design flow. As a designer you want to see stability and incremental improvements to the IP you are using over time. The last thing you need just before tape-out is a major redesign of an IP or a library. If your supplier promised to enhance the .lib files with improved power arcs you don't want to see different setup/hold conditions or different port locations in your LEF. Those point to major changes to either the GDS or the characterization flow that was used, therefore posing a risk for the synthesis flow of your design. With Crossfire-diff you catch these things right-away and you can use the report to have a discussion with your supplier. Perhaps they had very good reasons for making the changes that they did. The changes as reported by Crossfire-diff give a good handle for discussing all the relevant aspects and give you back the confidence you need to release this IP into your design flow.

Q: So this means Cdiff has handles to filter on differences-of-interest?

Johan: This is something we're still working on. Currently all differences are reported, customers are able to filter out the stuff they don't want to see. This we feel is an important usage model: let Crossfire always run to the full extent and apply a filter on the result so that

you are able to review everything. If you'd limit Crossfire from detecting certain issues, you might be fooled in believing a partial result to be a full result. Thus never having partial difference results prevents for a lot of "false positives", as we call them.

Q: Is there any difference in the issues that Cdiff reports vs the complete Crossfire checks?

Johan: Not really, Cdiff has no limitations in that respect. It will report on pins or timing arc conditions just like Crossfire does. Also pin shapes in LEF and cell functionality are compared using Cdiff. You do need to bear in mind that the checks in Crossfire constitute a lot more than just the difference from a golden reference view. Properties like pin routability, or cell-delay monotony through temperature establish a measure of quality, but are not captured by Cdiff.

Q: A popular use of the existing diff is in the "blame" functionality of design-repositories, do you see a role for Crossfire-diff there as well?

Johan: This is clearly the next step. As Cdiff is conceptually identical to a regular diff, you can use the two interchangeably. Use textual diff if you care about formatting, and Cdiff if you don't, or for binary views. A first step is to simply hook Cdiff into the repository of choice. Repository vendors typically provide plugins that allow for such customization. You can then imagine scenarios where the IP project manager receives an automatic email on, say, a new power-pin checked into the repository when the library is supposed to be in sustaining mode.

Q: A first step implies also next steps…?

Johan: Sure, but we're still investigating this. To save on database size, repositories typically only record the changes when going from one revision to the next. If you'd calculate those changes using Crossfire-diff, you'd save tremendously on the amount of changes you need to store and at the same time obtain a standardized textual representation of all your database views. Anything you didn't change cannot contain new bugs. The Crossfire technology is all about leveraging this concept, I feel we have a lot more potential here.

Conclusion

Cdiff is clearly more than a re-packaging of the same technology. It allows users to directly obtain first-level answers when new IP is received (Are the expected changes all there? No unexpected changes?). Even though an IP release may be perfectly consistent in itself, in the context of a maturing design flow, an unexpected large re-factoring is something SoC designers are understandingly weary off. This creative reincarnation of the common data-model technology that underlies Crossfire, convincingly shows the potential for a tighter integration of the Crossfire technology with SoC design management tools.