Correct by Construction Design Using Crossfire

A Fractal Whitepaper

Introduction

State of the art SoC design needs to rely on a correct-by-construction principle for the design creation flow. The increasing number of components from different suppliers and the amount of process corners that need to be covered lead to an explosion in both data-volume and data-variety. Crossfire can help in achieving a working design within a predictable flow by ensuring that every design step is validated and every imported IP-component is qualified. The different positions where Crossfire can anchor design flow quality are presented in this paper as QA related Frequently Asked Questions.

How can Crossfire be inserted in a Library Characterization flow?

Obviously cell-library characterization targeting any advanced process node will need to be correct-by-construction. One cannot possibly rely on trial and error or leave validation of the results as a final step in the characterization flow – to be skipped when time pressure is mounting. The increasing amount of variability that needs to be taken into account is pushing the number of process corners characterization needs to cover into the 100's, impossible to cover by inspection or simplified tests.

Instead, every cell-model from every corner and for all modeling formats (NLDM, CCS, ECSM) covering timing, power and noise needs to be addressed for questions like:

- Have all arcs been populated with data?
- Do these numbers pass basic sanity checks?
- Does the set of arcs match those required by the cells' functional model?
- Does the characterization data exhibit the expected trends, e.g. vs capacitance, temperature?

To perform such checks during the characterization flow, Crossfire may be run in batch-mode as part of the library characterization flow. Using the GUI, the library design engineer defines the various target formats that need to be validated and defines a golden reference. This reference could be a Verilog model or a standardized cell-definition format that is input to the characterization, circuit synthesis and physical layout generation tools.

Dedicated Crossfire-checks validate the completeness of the timing arcs as described in template .lib files against the reference Verilog. Each .lib file which is then generated by the characterization flow is then verified against a single reference .lib for completeness and sanity. Finally, trends are checked by comparing delays, capacitances and resistances from all different process corners.

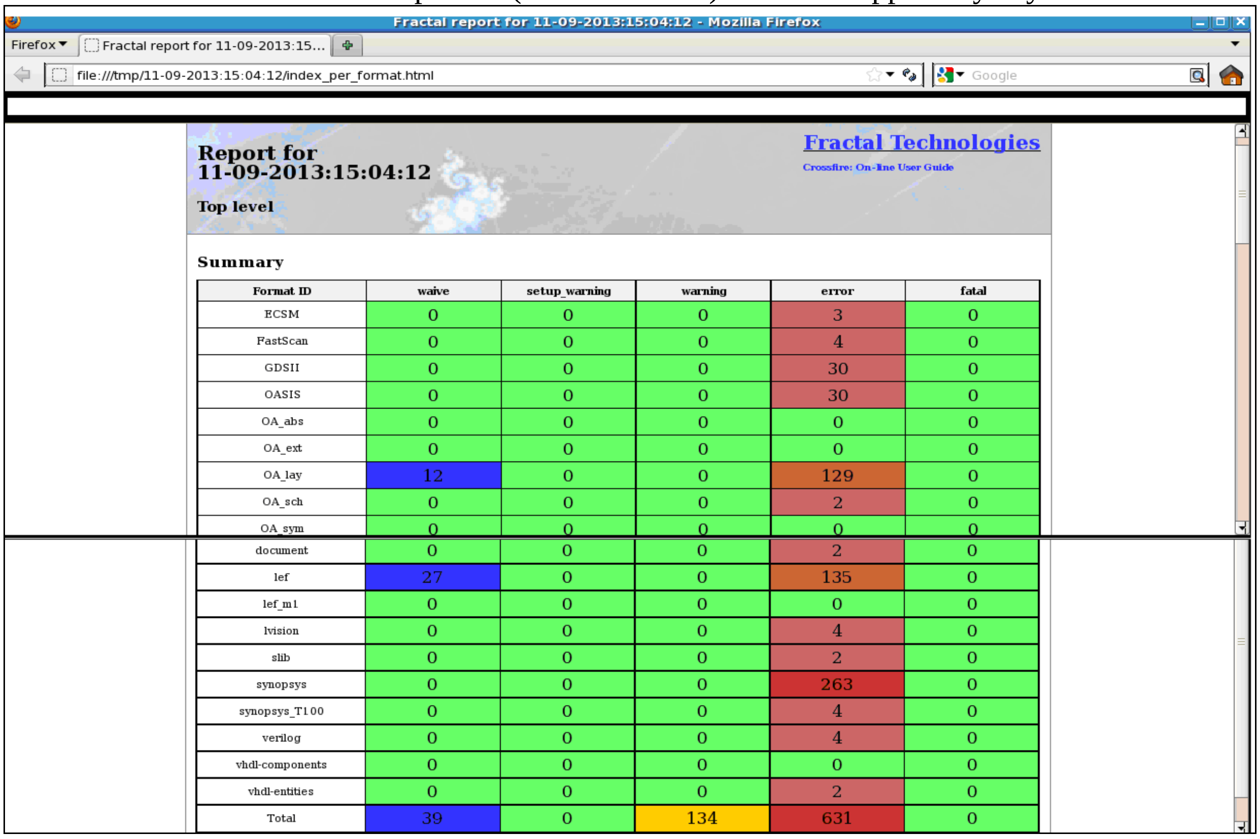
As the entire process is fully automated, the library characterization tool suite may be even instructed to respond to Crossfire-detected issues. An example of such a response is to restart certain SPICE simulations with higher accuracy settings if an expected delay-trend is not observed in one of the generated .lib files.

Having Crossfire embedded into the characterization flow provides confidence in the data: all results produced by the characterization flow are bound to be good and can be trusted. Never will a validation step have to be skipped before library shipment since all data produced is already inherently validated by construction.

Can Crossfire provide a sign-off solution for libraries?

Besides a package of data describing a well-characterized and documented library for the target process node, customers also expect the data to be already qualified. This allows designers to immediately start working with the latest library release once it becomes available, and to focus on their core strength: the design of circuits that meet the power and performance targets set for the design. If the library comes pre-qualified, designers don't have to spend valuable time and resources digging through the data to detect obvious flaws. But what's worse are the delays involved in the feedback loop back into the library provider to get them resolved. This can take significant hit on the already stressed SoC design schedule.

The solution is a well-documented, easy to read qualification report that Crossfire can provide. Crossfire results are essentially composed of a database of checks performed, together with easily accessible HTML reporting. This allows users to quickly browse through the quality checks performed, review what was found to be correct and – importantly – review what was found to be imperfect (waived checks) but was shipped anyway.


Fractal report for 11-09-2013:15:04:12 - Mozilla Firefox
Firefox
Fractal report for 11-09-2013:15...
file:///tmp/11-09-2013:15:04:12/index_per_format.html
Google
Fractal Technologies
Crossfire: On-line User Guide

Report for
11-09-2013:15:04:12

Top level

Summary

Format ID	waive	setup_warning	warning	error	fatal
ECSM	0	0	0	3	0
FastScan	0	0	0	4	0
GDSII	0	0	0	30	0
OASIS	0	0	0	30	0
OA_abs	0	0	0	0	0
OA_ext	0	0	0	0	0
OA_lay	12	0	0	129	0
OA_sch	0	0	0	2	0
OA_sym	0	0	0	0	0
document	0	0	0	2	0
lef	27	0	0	135	0
lef_m1	0	0	0	0	0
lvision	0	0	0	4	0
slib	0	0	0	2	0
synopsys	0	0	0	263	0
synopsys_T100	0	0	0	4	0
verilog	0	0	0	4	0
vhdl-components	0	0	0	0	0
vhdl-entities	0	0	0	2	0
Total	39	0	134	631	0

HTML Report: Top level report on per view basis

Crossfire has an extensive error waiving mechanism that goes way beyond suppressing errors under certain conditions. Such a basic suppressing-mechanism essentially leaves it to the library provider to decide what is correct and what is not, without providing any transparency to the library-user. Crossfire instead always presents the entire check-set in its report to the inspecting user who can then judge that indeed all checks that are required have indeed been run. Moreover, those that have been waived are still visible and can be reviewed. Here the motivation for the waive as created by the library provider provides the confidence to the library-user that the data can be trusted.

"It's a cellview Jim, but not as we know it"

Another useful item worth pointing out at this stage is the validation of the datasheets. Libraries require extensive documentation of their functionality, footprint and summarized timing/power characteristics. This data, often in HTML format, is treated by Crossfire as just another cell-view, similar perhaps to Verilog or a schematic. A documentation HTML view has terminals, a functional model, delay arcs, etc, just described in a different format. Crossfire parses these documentation views and compares them versus designated reference views to validate their contents. Thus, also the library datasheets obtain the same trust-level as the Verilog or CCS data.

I just received this IP block from my vendor – can I trust it at all?

Not all corners of the globe develop in the same pace so it just might be that your IP vendor makes deliveries that are not accompanied by a Crossfire check report and error database. In that case you either trust your vendor to do a good job, based on his eye-color or on a long standing relationship, or you decide to run some incoming inspection checks before you start using the data.

Crossfire is ideal for this purpose as it does not make any assumptions on the structure of the folder hierarchy or filenames of the input data. When starting up, all Crossfire needs is a pointer to the root folder of the data after which it will explore the entire directory tree for familiar file formats and databases. When these are found, also all relevant checks for these files are identified and selected to be run. So by default a comprehensive, relevant integrity check run can be started within seconds of unpacking the data.

Setting up a run-set for a checking tool easily is one thing, understanding the outcome of the checks is perhaps even more important. Crossfire results are presented as a hierarchical tree of checks where users can browse through categories to get a feeling for the areas where the incoming IP data is most wanting. The next step is to dive into individual checks to see that what is reported as an error is indeed an error (Crossfire Diagnose Option). By simply clicking on an error visualization the individual formats or databases are opened and the error, say a missing terminal, is highlighted. This is a great advantage for IP of which the structure is unfamiliar. It doesn't really matter where the .lib file or schematic database is located: Crossfire has found and can visualize the mismatches instantly – and very important without starting the EDA tools normally required opening files and databases.

Click here to watch Diagnose video

How can Crossfire assist during the IP design flow?

IP design is characterized by gradually lowering the level of abstraction describing the components, from specification down to functional, netlist and physical layout. Some of these steps result from automation (like synthesis), others from manual work. Yet consistency at each stage is a requirement – not only between the various block at a specific level but also between different hierarchy levels. One cannot simply introduce new terminals at netlist level as this would a priori invalidate a layout vs netlist check right before tape out.

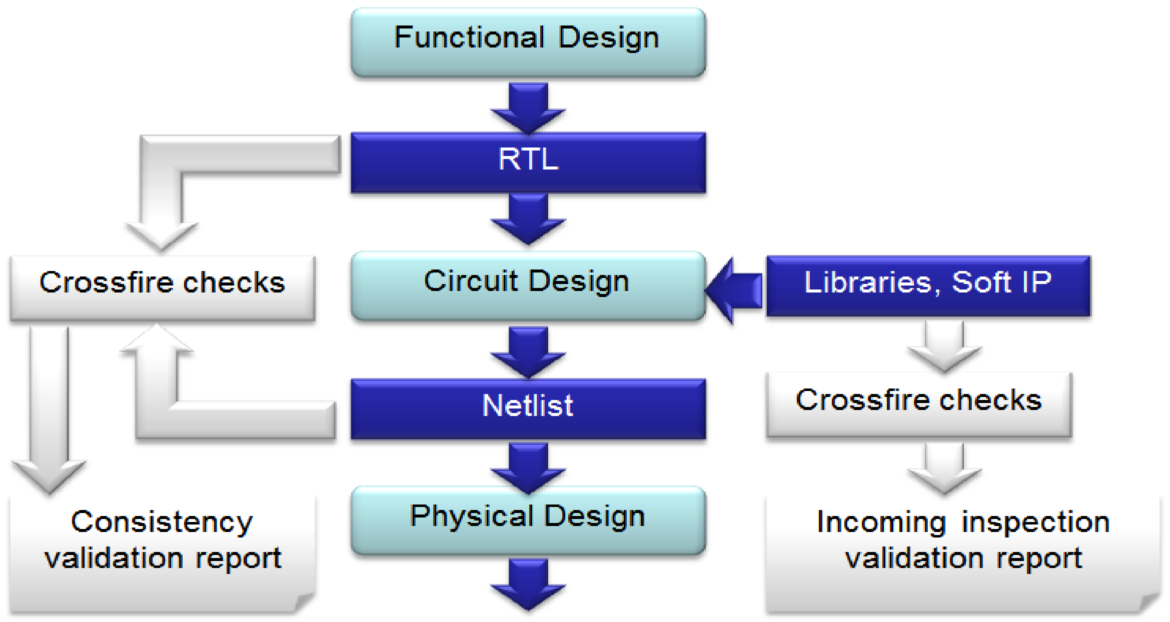


Figure 1, Crossfire usage inside the design-flow as well as for incoming inspection

Before a new version of a block or component is committed to the design repository, a Crossfire run on that result should be made mandatory. This way one can be ensured of compatibility of the new model with respect to the previous version. This ensures that any issue is detected early in the design cycle and is reported back to directly to the person responsible: the designer. The cost of correcting errors during the design is a multiple lower than the cost of repairing after design completion, right before the target ship date. This type of usage resembles the use of regression testing, a well established technique in software development.

Such QA checks can also have their place during the repeated build runs that are typically run every night to automate the design flow as much as possible. A prime component of a predictable, robust design flow is the automation of the entire design flow starting at a very early stage. During behavioral design, a full physical layout can be generated using black boxes as placeholders for the to-be synthesized components – that way ensuring that the layout synthesis of the remaining components can already be tested and verified. When more details become available, the black boxes may eventually be replaced with functional parts, which are then connected by an already proven and robust final place and route flow. During such a repeated automated synthesis flow, Crossfire may be activated to validate the outcome of the various stages against previous levels. This ensures that none of the automated tasks introduces any anomalies and that the final synthesized layout can be trusted to pass final DRC/LVS checks.

Conclusion

This paper has illustrated the opportunities for deploying Crossfire validation technology to obtain a correct-by-construction design flow. Because of its flexibility, Crossfire can contribute both during handoff of IP and cell libraries as well as during the design flows that create them. These different quality validation insertion points each require different tool properties. During the handoff of IP and libraries, a GUI and automatic detection of databases, file-formats and corresponding checks is key. During the design-flow, these same checks need to be activated in batch mode and driven from an API, where more emphasis is placed on automation, waving of rules and quickly identifying views and lines that are inconsistent.

The versatility of Crossfire allows it to be used throughout the entire design flow – so a consistent set quality checks can be deployed from the specification down to the final - correctly constructed - layout.