Setup Generation for Fractal Crossfire

A Fractal whitepaper

Introduction

A common misconception related with design tools is that they by magic auto-configure themselves to the needs of the designers that need to use them. Design tools undoubtedly provide a solution to a real need of a design organization. Be it simulation, synthesis, data-management or some flavor of design verification; these are all concepts without which a state-of-the art System-on-Chip cannot be efficiently and reliably designed. Hence software solutions have been created by the EDA industry that can effectively deal with these issues as proven by benchmarks, success stories and industry acceptance.

But these rewards as promised by design-tools don't come for free. Between the introduction and realization of the benefits of a design tool is the process of "setup creation" where the tool is configured to make it deal with the specifics of the design-style and CAD flow of the organization. This setup effort is a necessary consequence from on the one hand the differentiation of a design-organization (our designs are best because of their unique properties and design-style) and on the other hand the need to work with 3-rd party tool vendors (as fully customized internal tool development no longer pays off).

The success and usability of a design-tool highly depends on the effort spent in setup creation and tool customization. This is a task that is usually performed by key-users and CAD teams in close cooperation with the application-engineers of the tool vendor.

In this white-paper we will discuss this customization process for the Fractal Crossfire IP qualification tool. We will review the toolbox provided by Crossfire to automate the setup process and the ways in which a design organization can further leverage a well designed IP qualification setup by providing it as a standard to its suppliers.

IP qualification using Crossfire

Crossfire is the de-facto industry standard tool for IP qualification. In a typical usage scenario an SoC design group will receive IP from different suppliers for integration in a final design. Such IP deliveries will be of various types, ranging from libraries (standard-cells or IO-cells), Hard-IP blocks (SERDES, ADC, DAC, SRAM or hardened soft-IP) to Soft-IP blocks like CPU-cores.
Also the suppliers can be wildly different. Foundation IP like IO-cells or SRAMs is often supplied by the Foundry partner, other blocks can originate from third-party vendors or different design-teams within the same company. Yet all of them need to be qualified before they can be used reliably in a design for simulation, synthesis and verification.

On a very basic level at least the bill-of-materials needs to be checked: are all databases and models actually present? Are all cells represented in all models? Next the internal consistency between the various databases needs to be established, for example by asking whether all terminals from Verilog are present as pins in LEF? And do the delay and power arcs between

Liberty files match? Finally also trends between process corners and power-domains should be considered. This includes checks like: Do delay and power arcs trend consistently with temperature and supply-voltage? Do related power pins in Liberty match the connectivity in the SPICE models?

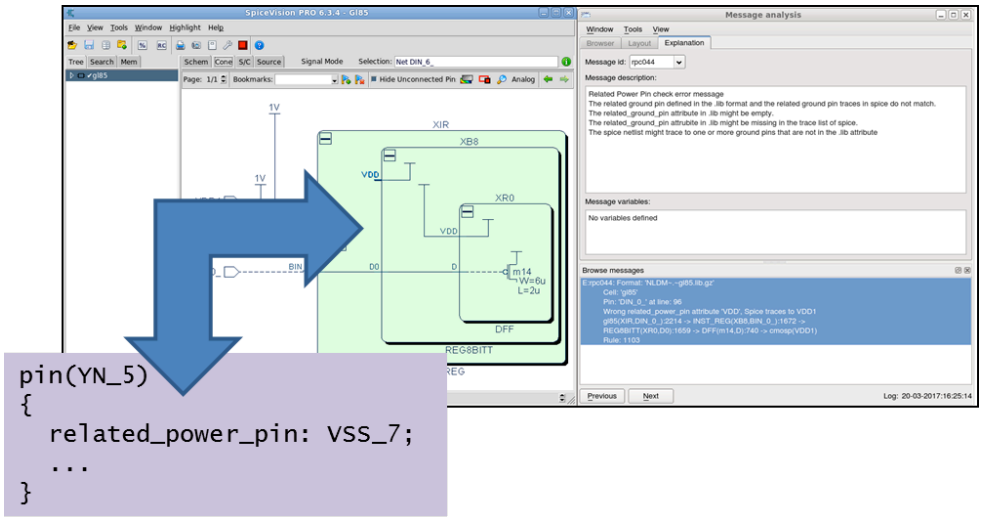

Figure 1, Crossfire results for related power pin check

Crossfire contains roughly 300 of such rules to assess the quality of an IP block. It is able to apply them on all known industry-standard databases, models and formats even including data-sheets. Because of its high level of internal parallelization Crossfire is able to run a complete qualification on a typical 4TB IP database within 4 hours on a high-performance cluster.

The result is an extensive HTML report and check-results database that allows reviewers to quickly zoom in on the root causes of any violations found. By simply double-clicking on the error, the related formats are opened automatically, highlighting the cells or lines that exhibit the issue in more detail. Reviewers can then easily decide to either report this back to the IP supplier, permanently ignore the issue by changing a check-specific tolerance or finally by temporarily waiving the error. The latter explicitly documents the acceptance of a specific rule-violation instance and, by making it time-bounded; one can ensure that such issues do not end up in a final design.

The advantages to the design-organization of having an IP qualification solution are obvious. All comes down to limiting the time between bug-insertion and detection. When IP internal consistency issues are already filtered out by Crossfire IP qualification, everything that is detected during final design verification is truly a design issue and no time is wasted by tracing back to problems within third-party IP. As a result, tape-out schedules become much more robust and predictable where it matters most: during final design verification for timing, power, noise and layout vs netlist consistency.

Setting up Crossfire

Clearly an industry standard IP qualification tool is a must for any SoC design organization. Now let's get back to the problem posed in the introduction: how does a Crossfire user get to harvest these benefits, with as little as possible overhead time spent in creating and maintaining setups?

An important principle of the Transport language that describes Crossfire setups for reaching this goal is that it is entirely based on generic concepts rather than IP-specific checks. This idea is best explained by means of an example. An example of a check, or a rule-instance, would be:

- *"The drive current of terminal Y of cell AN2DR3 as listed in the Liberty file cms07_T25C.lib file for (temperature 25 degrees) must be lower than the same current for cms07_T80C.lib by atleast an absolute margin of 0.05".*

Although this is something one clearly wants to verify in an IP, none of the elements in this check is portable to checking another IP that has only the slightest difference with the original IP. One would manually have to change cell-, terminal- and file-names. Easy ways out of such a setup black-hole are deceptive: one could decide to simply apply the check to any terminal in any Liberty file. Yet that would result in such a vast amount of false violations (as not every cell is a logic cell) that it becomes unworkable as an acceptance criterion.

Instead what is needed for repeated application over different IPs, but of similar category, and for publication as an IP certification standard, is a setup described as a set of generic rules, that are applied to categorized groups of objects such as cells, terminals, arcs and databases. The latter are known in Crossfire terminology as cell-, terminal-, arc- and format-classes. By applying such generalizations, the example check above can be transformed into a generic rule:

- The cell-name AN2DR3 can be replaced by a group of cells like "All Cells". This is known as a cell-class in Crossfire setup terminology
- The terminal to which the check is applied can be terminal-class called "All digital output terminals"
- The databases files cms07_T25.lib should be replaced by a class of formats like "Liberty files" which then includes both the CCS and ECSM variants.
- The 0.05 absolute margin finally can be exchanged with a variable such as 'Temperature_current_margin' which could have different values for different types of IP or technology nodes.

Once the generalizations as described above are applied we arrive at what is called the equivalent, generic "rule" in Crossfire terminology:

- *"The drive currents for all logic cell output terminals in Liberty should show an increasing trend between characterization temperatures by at least a fixed margin."*

Once a Crossfire setup is created based on such rules, it is then easy to refine it and apply to different IPs of the same type. All that is needed for (re-)application is to decide on the rules

to be applied, the appropriate values of the limits and the contents of the classes for cells, terminals, arcs and formats. The associated data flow is illustrated in the figure bellows;

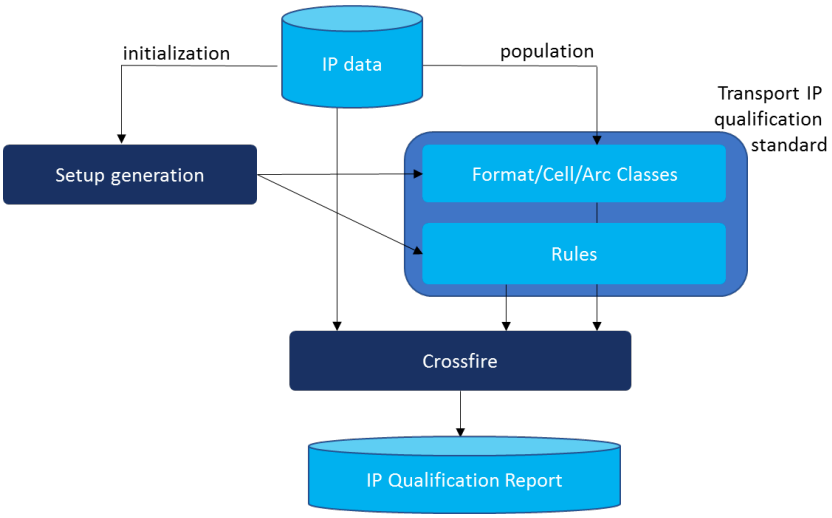

Figure 2, Crossfire setup creation data flow

As can be seen, a standardized IP rule-deck for Crossfire is composed of a set of rules that apply to classes of objects together with a set of rule-constraints. The Crossfire automatic setup utility "frc_create_cfs" will automatically create a first version of such a setup, consisting of a set of readable YAML files. This needs to be done only once for a specific category of IPs to be qualified. After the rule-constraints have been assigned meaningful values appropriate for the IP and technology node, a process explained below, all that is needed to apply the rules to a different IP is to re-populate the different cell-, terminal- and format-classes. This re-population is partly automated (Crossfire can classify e.g. all Verilog files it finds below an IP root folder), and partly relies consistent naming between IP releases.

Because such a rule-deck has now become IP-instance independent, Crossfire users now have the opportunity by using it as a standard to be applied to various IP blocks of the same category. Appropriate rule constraints may be identified for a particular process node, rule violations can be discussed with IP suppliers and either accepted or waived for the time being. This is where a standardized Crossfire setup becomes re-usable and can be deployed as part of system of best-practices that is continually improved with every new IP, technology node or IP-supplier.

IP-exchange through Transport

Transport is the name of the YAML based format in which standardized rules for IP qualification by Crossfire are described. Transport decks created and maintained by the mechanisms described above allow design-organizations to make sure that the IP they receive is internally consistent and match the requirements of the design-tools and design-style. An example of how a rule is described in Transport is presented in the following figure:

```
cell_presence:
       rule: [21, 22]
       enable: 'True'
       formats:
           - AllFormats
           - synopsys_T100
           - ECSM
       format parameters:
           DEFAULT:
               ignore: UNDEF
               ignore_classes: UNDEF
               mandatory: UNDEF
               ignore_instantiated_cells: 'False'
               mandatory_classes: UNDEF
           AllFormats:
               ignore_classes: IGNcells
               ignore_instantiated_cells: 'True'
               mandatory_classes: STDcells
           ECSM:
               ignore: [AOI22, BUF1, BUF4, DFFPC_S, DFF_S, TIEHI, TIELO]
               mandatory: AND2_X1
           synopsys_T100:
               ignore_classes: IGNcells
               mandatory: XOR2_T100
               mandatory_classes: STDcells
```

```
format classes:

  AllFormats:
     formats:
       files: *  # All files that hold design data

  AllLiberties:
     formats:
        type: synopsys  # All design data in liberty or *.lib files
```

Figure 3, Transport rule example

As a metaphor, a Transport rule set can be compared to a design-rule manual or DRC-deck as issued by a foundry. A Transport deck ensures that the IP received can be synthesized, simulated and verified by the design-group, just as a design-rule deck ensures that a GDS file can be OPC-d for creating a reticle that ensures working silicon with sufficient process window.

Crossfire customers also deploy Transport decks in a similar way as foundries use DRC-decks: Transport decks are sent to IP-suppliers in order for the supplier to perform the IP qualification before shipment. This narrows the gap between bug-creation and bug detection even further, especially when IP designers start to make use of Crossfire checking as part of their nightly regression runs.

How to develop a Transport deck

Getting started from scratch is the most difficult phase in the adoption of any new design-tool. The first step, to get to an initial setup that is both working and efficient to work with requires a bootstrapping process that involves training and often extensive on-site tool-vendor support. The auto-setup creation functionality contained in Crossfire as described above allows users to leapfrog to this initial working state at the push of a button. What is then required is a customization process of defining appropriate categories of cells, terminals and formats to which rules need to be applied and to decide on rule-constraints that are appropriate for the technology node.

On deciding on appropriate values for rule-constraints Crossfire offers support with the so-called error-fingerprint visualization. When deciding on rule-constraints, such as the rising current margin discussed above, one needs to find a value that filters out the real outliers that need further investigation, so the constraints should not be too relaxed, yet on the other hand does not report the entire library as a violation. As an example consider the following screenshot:

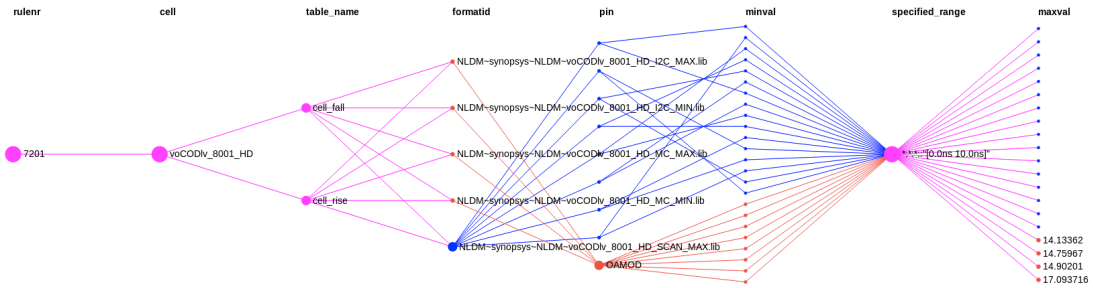

Figure 4, Crossfire error fingerprint visualization

This is the result of a range-check on the delay values for a subset of cells. As is instantly apparent, most pins of this design pass this check, except for the pin OAMOD, for which also the actual value that exceeds the range is shown. From a graph like this, appropriate rule-constraint values can easily identified for this combination of rule. IP and technology node.

In the example, the range check was found in violation for pin OAMOD and we could also conclude that that the initial range values are appropriate for the rest of the library. Now at this point it is not clear what these large delay values for pin OAMOD actually mean: was there a characterization issue that is showing up here or is there some good physical explanation for these large delay values? Deciding on this takes discussion with the characterization team of the IP supplier. Meanwhile Crossfire has the option to waive this rule specifically for this IP/Cell/Pin/Arc combination for a limited amount of time, say 1 month. After that, the waive will be dropped and a solution needs to be in place: either the characterization was modified to bring pin OAMOD in line with the rest of the library or there is good reason to exclude this particular category of pins from delay-range checking.

Conclusion

For an IP qualification tool like Crossfire there is immense value in the setup itself. A Crossfire setup, called a Transport deck, is a representation of the standards to which IP needs to comply before it can be inserted into a design-flow. By treating it as a standard, that is by discussing the content in standardization groups involving IP suppliers and SoC designers, Transport decks are continuously improved and become all the time better suited to the category of IP they are created for. Crossfire users, either IP users or suppliers, benefit from this by seeing only relevant violations, no false violations, while at the same time being ensured that checks that pass are indeed reliable and do not contain false positives.

The automatic Transport setup generation and the strict use of generic concepts like cell-, terminal-, arc- and format-classes ensures that Crossfire users can quickly see the benefits after tool introduction, while applying setup creation resources only where it adds value to the organization: in the selection of categories and rule-constraints that embody the unique strengths of the design-organization.