Towards a Quality-Aware Design Flow

A Fractal Whitepaper

This note addresses the need for IP developers to have QA validation solutions fully integrated within their design flow. Failing the complete embedding of QA checks and procedures in the daily work of designing and enhancing IP blocks, achieving compliance with quality standards will always remain an afterthought: something that is last on the to-do list before a new IP version can be shipped to customers. And consequently, something that can be short-cut or skipped should circumstances so dictate.
Next, we will review the different insertion points for QA tools, and in particular Fractal Crossfire, within a design flow and compare those to the necessary levels of integration and user-friendliness.

Incoming inspection

The traditional position of QA solutions within an IP-based SoC design flow is at the stage of incoming inspection. The IP integration team receives IP blocks typically from several different suppliers such as foundry, internal design-groups, and third party IP suppliers. Some of these IPs have been specifically designed towards integration with the current SoC design, others are re-used from different ASICs or are general-purpose and have been designed to fit the needs of many different users. It has long been recognized that many design re-spins and tapeout delays can be traced back to incompatibilities of incoming IP with the design-flow or within the IP itself. Thus, the ability to detect such mismatches as early as possible is crucial for a predictable SoC design flow that does not suffer from schedule setbacks because of late bug-fixing. For example, if an inconsistency between the logic model and the abstracted layout views is detected only at the final chip-assembly routing-stage, the feedback loop need for repair is a long loop indeed.

Incoming inspection QA tools aim at shortening this feedback loop. They do so through an intuitive GUI that gives the IP-receiver an instant overview of what databases, formats and views have been received in the IP shipment, and the checks that may be applied to those views. For this level of QA awareness in the design flow, validation is an explicit step that requires an explicit, friendly tool that guides the user through the multitude of received data files and is able to pinpoint possible issues and trace those right back to the individual lines in files or cellviews.

Crossfire serves this type of usage by a well-organized GUI that allows users to cross-link the various formats and databases and the possible checks that may apply to them. Time-to-result is drastically shortened by Crossfires' capability to automatically recognize and group all databases and files that are provided for a given IP starting from a designated root-folder. Once the results of the QA checks are available, Crossfire provides graphical highlighting of mismatches within the different views simply by double-clicking individual error messages. Generic checks are known to often generate many correct but irrelevant view-mismatches. Crossfire allows users to focus on relevant mismatches only by waiving those errors that are deemed not applicable or fatal for the SoC design at hand.

Next step: no errors during incoming inspection

From the above explanation it is obvious that an explicit design-step (incoming inspection) needs an explicit, interactive tool to do the job (like the Crossfire GUI). Organizations that wish to reach a higher QA maturity level however, will feel the need to go further by making quality assurance an integral part of the design flow, rather than an explicit step. As a result, an incoming inspection, or sign-off check, will be clean by definition, thereby reducing the bug feedback loop even further.

This level of intimacy is, strange-enough, not aided by explicit, friendly GUI's that perform the necessary validation. Instead, the validation needs to run in the background, silently, and only request the designers' attention when a mismatch is found. Only for relevant, new mismatches, re-validation by the designer is required and a quick graphical check is to be performed by the user.

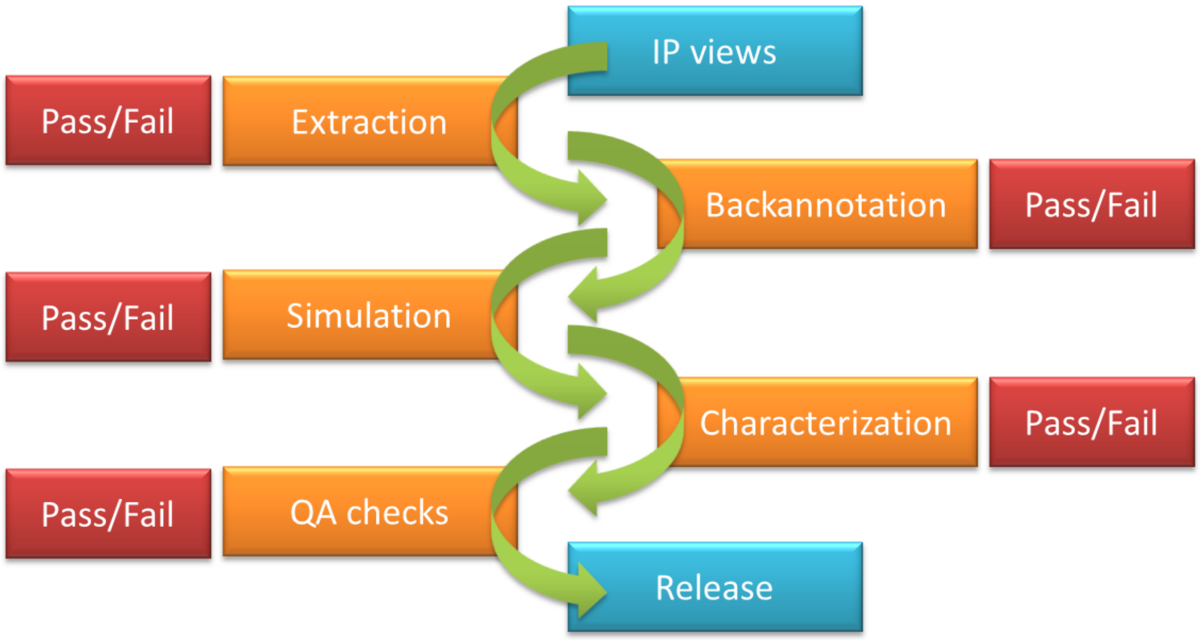

Figure 1, Automated development/characterization flow: all steps are push-button – so should the QA solution be.

Crossfire enables this integrated mode of operation through the Crossfire Setup API. The Crossfire Setup API allows users to create dedicated checking scripts in either Python, Tcl or Perl, that automatically locate newly changed databases and apply relevant checks. With the Crossfire Setup API, users can define databases, formats as they are present in the design-environment, possibly distributed over different network drives. Within these databases,(sets of) cells and views can be defined and relevant checks can be selected from the Crossfire archive of predefined checks, very similar to how checks would be set-up interactively in the Crossfire GUI. Following the setup phase, the Crossfire Setup API also contains routines for exporting the setup to the Crossfire GUI and to execute the checks directly from the Setup API script.

As the entire Crossfire functionality is now available within the flexibility of a programming language, the Crossfire Setup API allows far more flexibility and tailoring towards the IP at hand than is possible through any graphical user interface. Consider as an example the integration of QA validation with the design repository. The script that uses the Crossfire Setup API may consult the repository for those views that have changed in the working copy and only run the checks applicable to those changed views. If desired, such a Crossfire Setup API check script may be called as part of the repository commit-procedure. The results of the checks are as always contained in a database that can be later reviewed using the Crossfire Diagnose tool.

By deploying the Crossfire Setup API, QA checking has become completely integrated within the IP design flow and is no longer a separate activity. QA checking has now become more analogous to for instance simulation-verification in the design flow: when a design-change is made, a re-run of the vectors is a fully automated next step, which controls whether the new revision can be checked in and passed on the integration team. The actual simulation will be done completely in batch mode, and only when a vector fails will the designer open up the waveform viewer to trace back the root cause. Likewise with a Crossfire Setup API script, all checks run in the background, only requiring attention from the designer when a check has failed, at which point the full power of the Crossfire Diagnose tool is available to find the root-cause.

Conclusion

We have shown how a fully integrated QA solution allows an extra step to be made in shortening the QA feedback loop. The initial gain in becoming quality-aware is in having a dedicated, interactive QA solution available that can be used to inspect IP databases and formats and can assess compatibility of the IP with quality standards. The next step is creating IP that automatically passes standards compliance checks. This requires a QA solution that is embedded into the design flow and can be directly triggered by every relevant design-change. For an IP-designer, QA check results should be received like an RSS feed: only when available and relevant. The Crossfire Setup API allows design organizations to reach this next level of QA aware design flows.