

Quality Assured SoC Design Using Crossfire

A Fractal whitepaper

Introduction

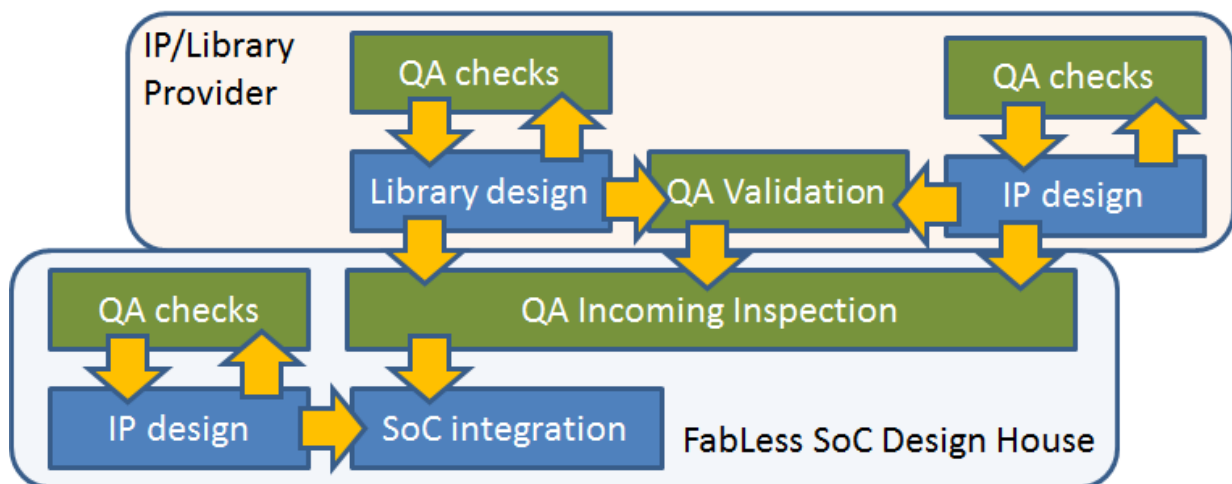
There is no industry where the need for early bug-detection is more paramount than in SoC design. Consequences like design-re-spins missed tape outs and hence missed market opportunities make the cost of late bug-detection prohibitive. Where earlier generations of SoC designs could be crafted by a team of limited size that could oversee the entire design process, design in the latest process nodes requires a different strategy. Designer productivity is lagging behind Moore's law that drives the increase of transistor density. Thus design teams are becoming larger and are comprised of multiple groups spread over the globe. Outsourcing of design-tasks by integrating third-party IP is mandatory to get the job done but reduces oversight of the SoC design process and leaves SoC design companies at the mercy of the quality strategy implemented by their suppliers. At the same time modelling of new physical effects like current-driver models for CCS and ECSM models needs to be taken into account.

It is clear that QA needs to be a shared responsibility by all partners in the SoC design flow, from library and IP providers to foundry and SoC integrators. Each of these partners needs an integrated QA solution in their part of the design flow; never should QA be an afterthought to be checked off right before IP delivery.

This paper describes how the Fractal Crossfire product can address these QA challenges. We will show how Crossfire has its place in every stage of the SoC design flow, serving different purposes to different users. Because of the breadth of deep-submicron quality checks and the rich set of supported EDA formats and databases Crossfire can cover any design flow and quality requirements.

The SoC ECO System

The figure below depicts the typical situation of a Fabless SoC design house: the design process being a mixture of custom, in-house, IP design and integration of IP blocks acquired from third party vendors. Also cell and IO libraries are externally sourced and need to be checked for integrity and compatibility with the chosen design flow.



The boundaries as depicted are more or less arbitrary. An IDM, for instance, may develop all these components in-house. The other end of the spectrum consists of SoC designers that only integrate external IP and put all their added value in the software that runs on their device. What are common though are the four categories of QA checks:



- **On-going QA checks** throughout the design process to ensure that the creator of the IP or library cells detects any issues while still modifying the IP. This ensures the smallest overhead for QA in the entire design flow by having the issues fixed by the best-qualified expert – the original designer.
- **A validation checks** are done at the end of a design-task, before shipping IP or cell-library to the customer. These checks involve more integration style checks involving compatibility between different parts and models of the deliverable.
- **QA Incoming Inspection** is again required at the receiving end and in part duplicates the checks already performed by the supplier. Their inspection's main purpose is to validate compatibility of the delivered component with the tools in the SoC design flow.
- **QA Requirements Forwarding** is the process where an IP consumer draws up a QA check set to be implemented by the IP provider, thereby greatly reducing the need for an elaborate incoming inspection task.

It should be noted that in-house activity these tasks are no different. Design-teams, often in different divisions and geographical locations can be more “alien” to each other than an IP supplier that is tightly linked with the SoC design company.

QA Aspects of Cell Library Design

The main characteristic of cell library is volume: the large amount of individual cells (easily reaching into 1,000's) and the large amount of different models to be created for them. Not only is Crossfire capable of dealing with this large variety of different models and checks, at least as essential is its capability of presenting the results in an easy-to-understand and accessible format, and its options for automating regression-style checking.

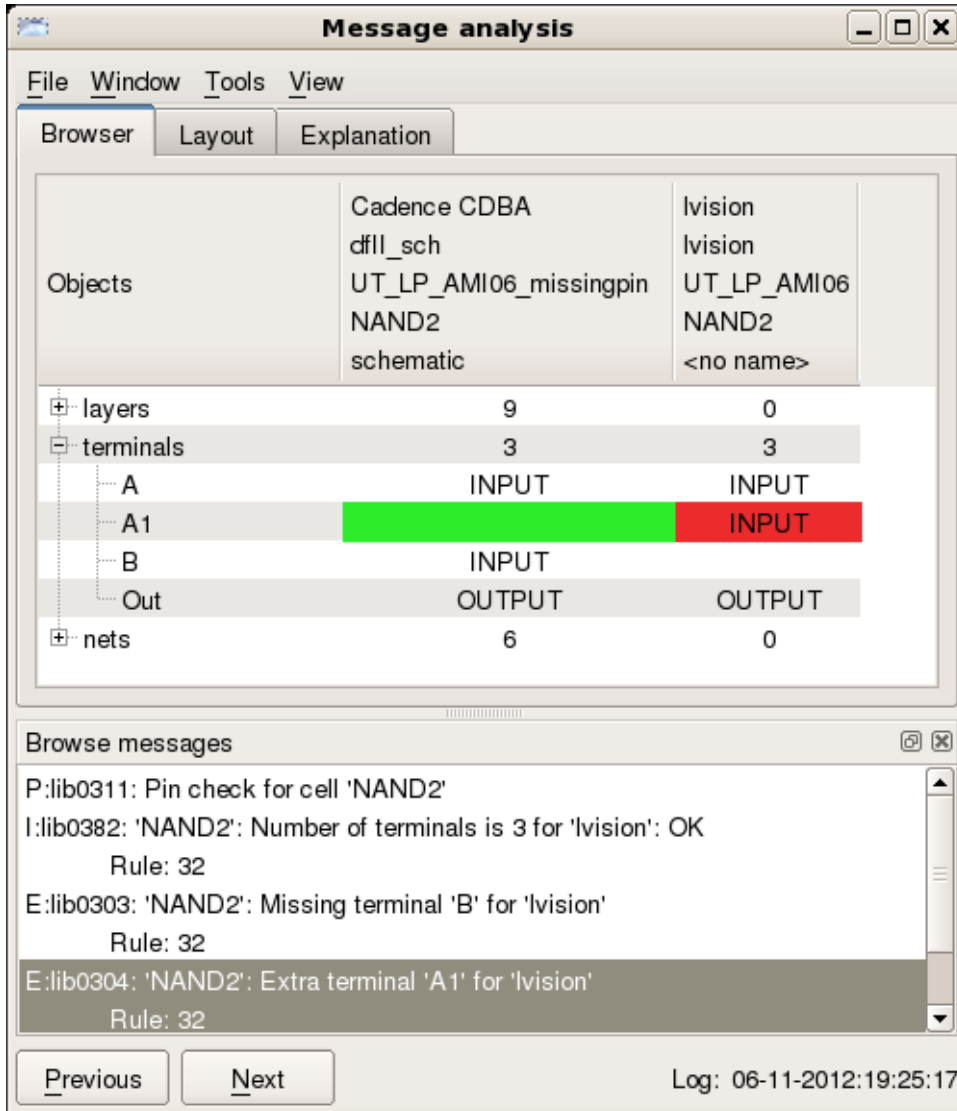
To elaborate on the rich set of input formats, the following formats are supported:

- Cadence DFII layout & schematic views
- Open Access layout & schematic views
- Milky-Way CEL, FRAM & CON views
- Verilog
- SystemVerilog (*)
- Verilog AMS (*)
- Tetramax
- VHDL
- Liberty NLDM,NLPM
- Liberty CCS, CCSN
- Liberty ECSM
- PLIB
- TLF (Timing Library Format)
- LEF
- DEF
- SLIB
- GDSII
- Oasis
- SPICE, HSPICE, CDL
- Fastscan
- STIL/CTL (Core Test Language)
- PDF (**)
- HTML
- All ASCII user Defined Formats

(*) No functionality check for this format

(**) After PDF to TEXT conversion

Crossfire internally constructs a unified data model covering all of these formats, making sure that the different cell models are all equivalent is a straightforward task. Any mismatch, reaching from mismatches on simple terminal names to conditional timing arcs and Boolean output terminal functionality are captured and flagged for the end-user with a graphical illustration where possible.



The screenshot shows the 'Message analysis' application window. It has a menu bar with 'File', 'Window', 'Tools', and 'View'. Below the menu bar are three tabs: 'Browser', 'Layout', and 'Explanation'. The 'Browser' tab is active and displays a comparison of two data models. The top part of the browser shows a table of objects:

Objects	Cadence CDBA	Ivision
	dfll_sch	Ivision
	UT_LP_AMI06_missingpin	UT_LP_AMI06
	NAND2	NAND2
	schematic	<no name>

Below this table is a tree view with the following items:

- layers: 9 (left), 0 (right)
- terminals: 3 (left), 3 (right)
 - A: INPUT (left), INPUT (right)
 - A1: INPUT (left), INPUT (right) - This row has a green highlight on the left and a red highlight on the right.
 - B: INPUT (left), (right)
 - Out: OUTPUT (left), OUTPUT (right)
- nets: 6 (left), 0 (right)

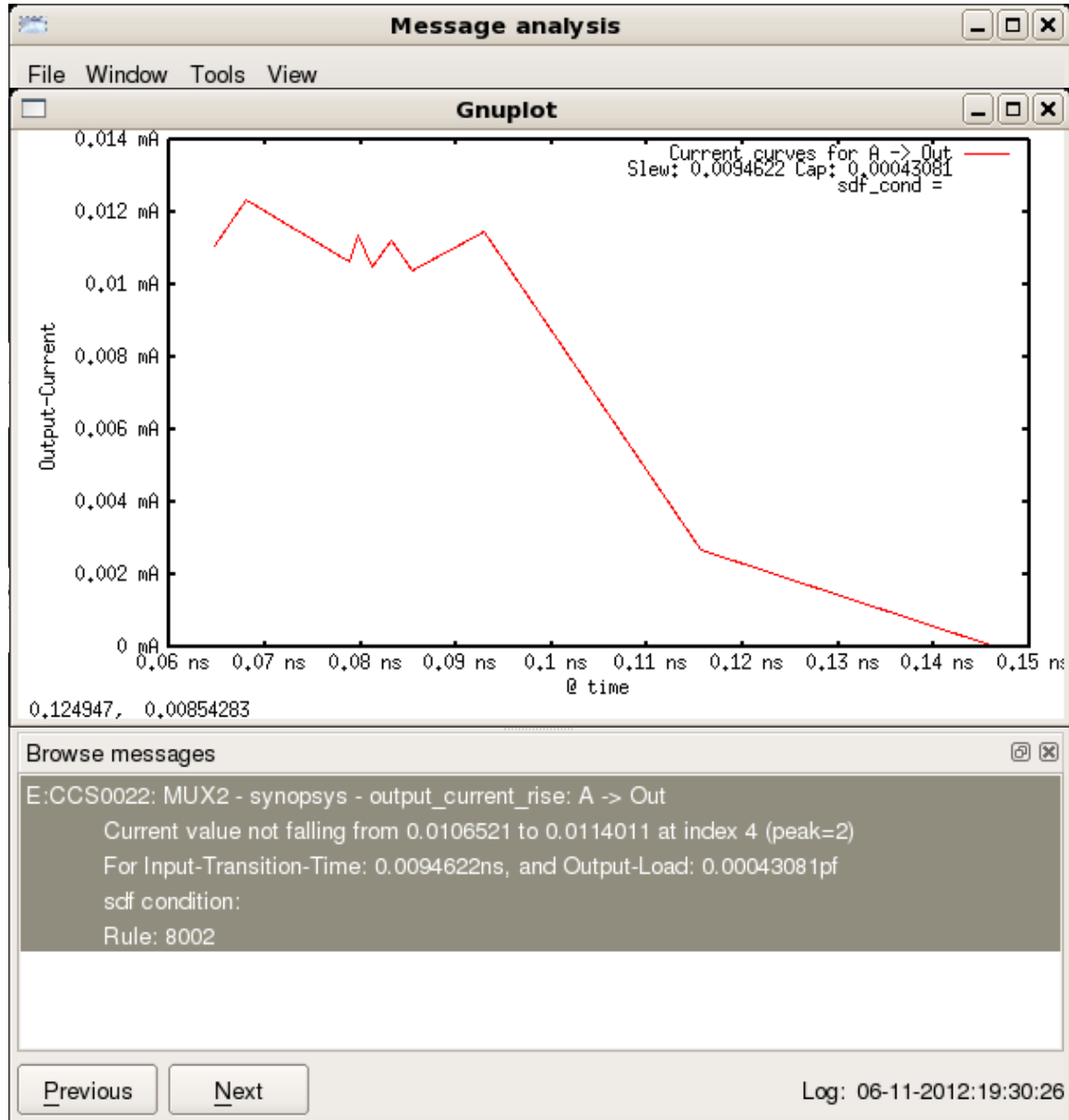
At the bottom of the browser is a 'Browse messages' section with a scrollable list of messages:

- P:lib0311: Pin check for cell 'NAND2'
- I:lib0382: 'NAND2': Number of terminals is 3 for 'Ivision': OK
Rule: 32
- E:lib0303: 'NAND2': Missing terminal 'B' for 'Ivision'
Rule: 32
- E:lib0304: 'NAND2': Extra terminal 'A1' for 'Ivision'
Rule: 32

At the bottom of the window are two buttons: 'Previous' and 'Next', and a log timestamp: 'Log: 06-11-2012:19:25:17'.

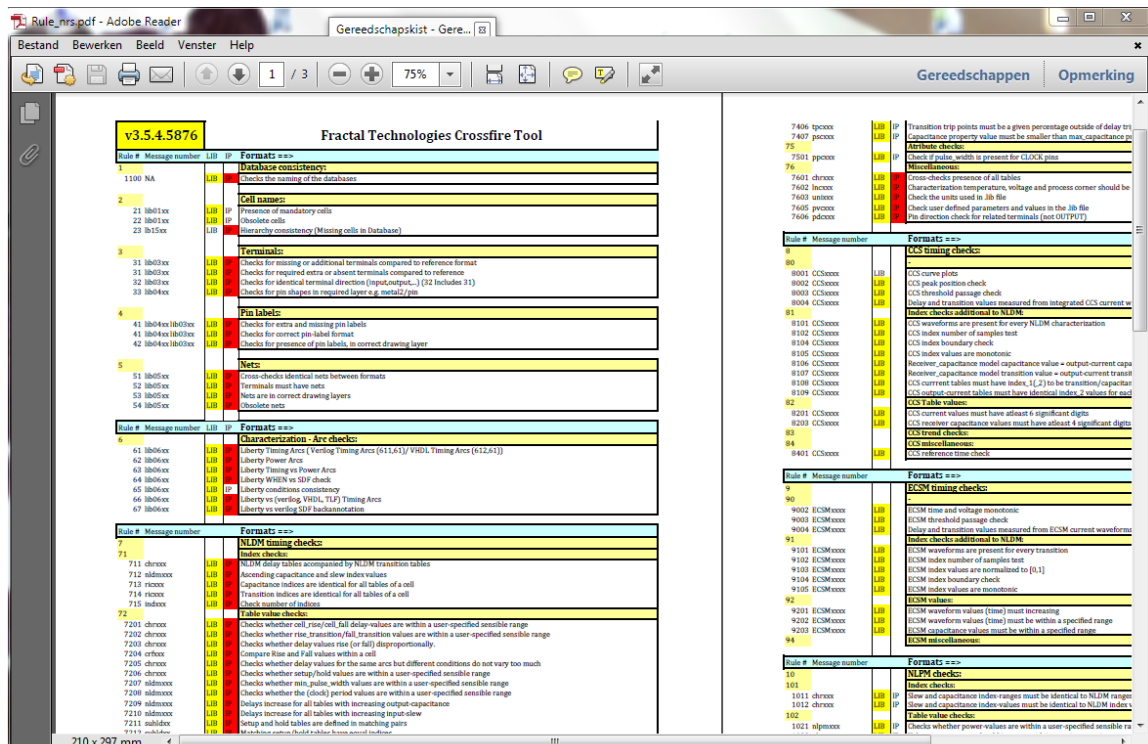
The unified data model allows users to also feed proprietary data formats into Crossfire for checking. Examples include library datasheets and proprietary cell-specification formats that describe pins, function, and number of tracks as input to cell-synthesis and characterization design flows.

The comprehensiveness of checks has already been mentioned and may be illustrated from the following screenshot from the CCS-specific checks.



For CCS and any other format all checks required to validate a cell-library are provided with the tool out-of-the-box. This allows users to quickly configure a suitable test set that covers the essentials of a particular database. Where necessary, the Crossfire API allows users to add their own customized checks to the check-set.

Of course, ease-of-use and automation are essential for a successful adaptation of Crossfire in any cell-library QA methodology. Ease-of-use for end-users has already been illustrated above; as important is that end-users are supported by partial check-sets suited for the different design-stages. Such partial check-sets are created by the CAD support team. Crossfire assists here with the automatic recognition of the tests necessary for each format. This allows the entire QA task to be sub-divided to accompany each completed step in the cells' creation process. This avoids needless backtracking, for instance by first checking on the pin-compatibility of layout, schematic and Liberty formats before starting lengthy SPICE simulations for creating ECSM models.



The screenshot displays the Fractal Technologies Crossfire Tool interface. The main window shows a list of rules categorized by format (e.g., Database consistency, Pin labels, Nets, Characterization - Arc checks, NLDWM timing checks). On the right, a detailed list of error messages is shown, including transition trip points, capacitance property values, and various timing checks (NLDWM, ECSSM, NLPPI).

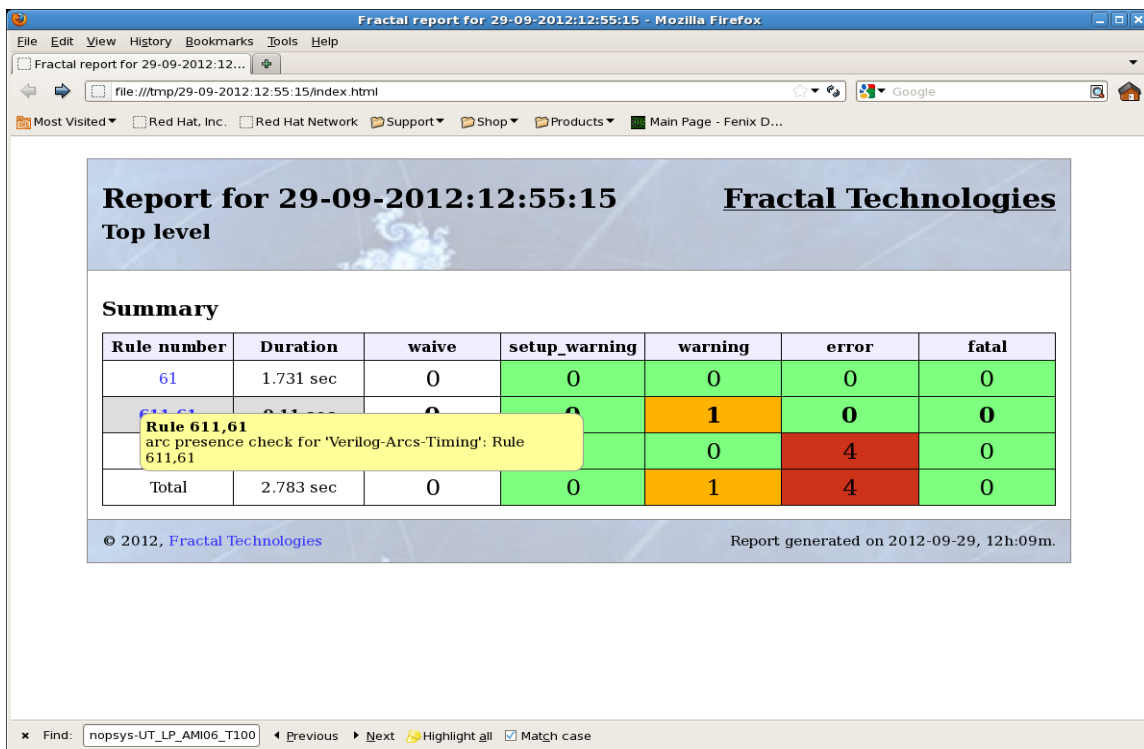
Rule #	Message number	LIB	IP	Format	Description
1100	NA	LIB		Database consistency	Check the naming of the databases
21	lib01xx	LIB		Cell names	Presence of mandatory cells
22	lib01xx	LIB		Cell names	Obsolete cells
23	lib15xx	LIB		Pin labels	Hierarchy consistency (Missing cells in Database)
31	lib03xx	LIB		Terminals	Checks for missing or additional terminals compared to reference format
32	lib03xx	LIB		Terminals	Checks for required extra or absent terminals compared to reference
33	lib03xx	LIB		Terminals	Checks for identical terminal direction (input/output... (32 includes 31)
34	lib03xx	LIB		Terminals	Checks for pin shapes in required layer (e.g. metal2/ps)
41	lib04xx	LIB		Pin labels	Checks for extra and missing pin labels
42	lib04xx	LIB		Pin labels	Checks for correct pin-label format
43	lib04xx	LIB		Pin labels	Checks for presence of pin labels in correct drawing layer
51	lib05xx	LIB		Nets	Cross-checks identical nets between formats
52	lib05xx	LIB		Nets	Terminals must have nets
53	lib05xx	LIB		Nets	Nets are in correct drawing layers
54	lib05xx	LIB		Nets	Obsolete nets
61	lib06xx	LIB		Characterization - Arc checks	Liberty Timing Arcs (Verilog Timing Arcs (41.1.6.1), VRDL Timing Arcs (41.6.1))
62	lib06xx	LIB		Characterization - Arc checks	Liberty Power Arcs
63	lib06xx	LIB		Characterization - Arc checks	Liberty Timing vs Power Arcs
64	lib06xx	LIB		Characterization - Arc checks	Liberty WREN vs SDF check
65	lib06xx	LIB		Characterization - Arc checks	Liberty conditions consistency
66	lib06xx	LIB		Characterization - Arc checks	Liberty vs Verilog, VRDL, TLFJ Timing Arcs
67	lib06xx	LIB		Characterization - Arc checks	Liberty vs Verilog SDF backannotation
71	lib07xx	LIB		NLDWM timing checks	Index checks
72	lib07xx	LIB		NLDWM timing checks	ECSSM delay tables accompanied by NLDWM transition tables
73	lib07xx	LIB		NLDWM timing checks	Ascending capacitance and slew index values
74	lib07xx	LIB		NLDWM timing checks	Capacitance indices are identical for all tables of a cell
75	lib07xx	LIB		NLDWM timing checks	Transition indices are identical for all tables of a cell
76	lib07xx	LIB		NLDWM timing checks	Check number of indices
77	lib07xx	LIB		NLDWM timing checks	Table value checks
78	lib07xx	LIB		NLDWM timing checks	Checks whether cell_rise/ceil, fall delay values are within a user-specified sensible range
79	lib07xx	LIB		NLDWM timing checks	Checks whether rise, transition/fall, transition values are within a user-specified sensible range
80	lib07xx	LIB		NLDWM timing checks	Checks whether delay values rise (or fall) disproportionately
81	lib07xx	LIB		NLDWM timing checks	Compare Rise and Fall values within a cell
82	lib07xx	LIB		NLDWM timing checks	Checks whether delay values for the same area but different conditions do not vary too much
83	lib07xx	LIB		NLDWM timing checks	Checks whether setup/hold values are within a user-specified sensible range
84	lib07xx	LIB		NLDWM timing checks	Checks whether min, pulse, width values are within a user-specified sensible range
85	lib07xx	LIB		NLDWM timing checks	Checks whether the (clock) period values are within a user-specified sensible range
86	lib07xx	LIB		NLDWM timing checks	Delays increase for all tables with increasing output-capacitance
87	lib07xx	LIB		NLDWM timing checks	Delays increase for all tables with increasing input-slew
88	lib07xx	LIB		NLDWM timing checks	Setup and hold tables are defined in matching pairs
89	lib07xx	LIB		NLDWM timing checks	Reference values hold tables from second instance
90	lib08xx	LIB		ECSSM timing checks	ECSSM time and voltage monotonic
91	lib08xx	LIB		ECSSM timing checks	ECSSM threshold passage check
92	lib08xx	LIB		ECSSM timing checks	Delay and transition values measured from ECSSM current waveform
93	lib08xx	LIB		ECSSM timing checks	Index checks additional to NLDWM
94	lib08xx	LIB		ECSSM timing checks	ECSSM waveforms are present for every transition
95	lib08xx	LIB		ECSSM timing checks	ECSSM index number of samples test
96	lib08xx	LIB		ECSSM timing checks	ECSSM index values are normalized to [0,1]
97	lib08xx	LIB		ECSSM timing checks	ECSSM index boundary check
98	lib08xx	LIB		ECSSM timing checks	ECSSM index values are monotonic
99	lib08xx	LIB		ECSSM timing checks	ECSSM values
100	lib08xx	LIB		ECSSM timing checks	ECSSM waveform values (time) must increase
101	lib08xx	LIB		ECSSM timing checks	ECSSM waveform values (time) must be within a specified range
102	lib08xx	LIB		ECSSM timing checks	ECSSM capacitance values must be within a specified range
103	lib08xx	LIB		ECSSM timing checks	ECSSM miscellaneous
104	lib08xx	LIB		ECSSM timing checks	ECSSM reference time check
105	lib09xx	LIB		NLPPI checks	NLPPI checks
106	lib09xx	LIB		NLPPI checks	Index checks
107	lib09xx	LIB		NLPPI checks	Rise and capacitance index-ranges must be identical to NLDWM ranges
108	lib09xx	LIB		NLPPI checks	Fall and capacitance index-values must be identical to NLDWM index
109	lib09xx	LIB		NLPPI checks	Table value checks
110	lib09xx	LIB		NLPPI checks	Checks whether power values are within a user-specified sensible ra

QA Aspects of IP Design

IP designers can rely on Crossfire for checking consistency and compatibility of their IP models similar to library developers. Notable differences are of course the size of the data (one large block versus many small ones) and the different types of models. Crossfire is optimized to deal also with large GDS or Verilog files, and is able to check compatibility with the different language dialects used for modelling IP such as Verilog-A.

IP can be delivered in many different ways, for instance as synthesizable digital IP blocks or as a hard analog macro GDS file. For each type, different check-sets make sure that Crossfire only checks those requirements that actually make sense. Routability checks for instance (i.e. are pins on grid and can they be reached by DRC-clean tracks from the IP boundary) apply to the various layout-related views only (GDS, OASIS, Milky-Way, OA and dfII databases). For an RTL-level IP, compatibility with the target design flow can be checked by running small fragments of the design-flow as part of the Crossfire check set. Only by running the original target synthesis or analysis tools to be used by the IP customer can we be sure that the delivered IP description is indeed valid input to the design-flow.

Crossfire also has a role to play in the final delivery of the IP to the customer. The QA reports generated by Crossfire in HTML format can be directly delivered to the IP-customer to demonstrate the integrity of the delivered IP block. An example of such a report is shown in the following figure:



Report for 29-09-2012:12:55:15 **Fractal Technologies**
Top level

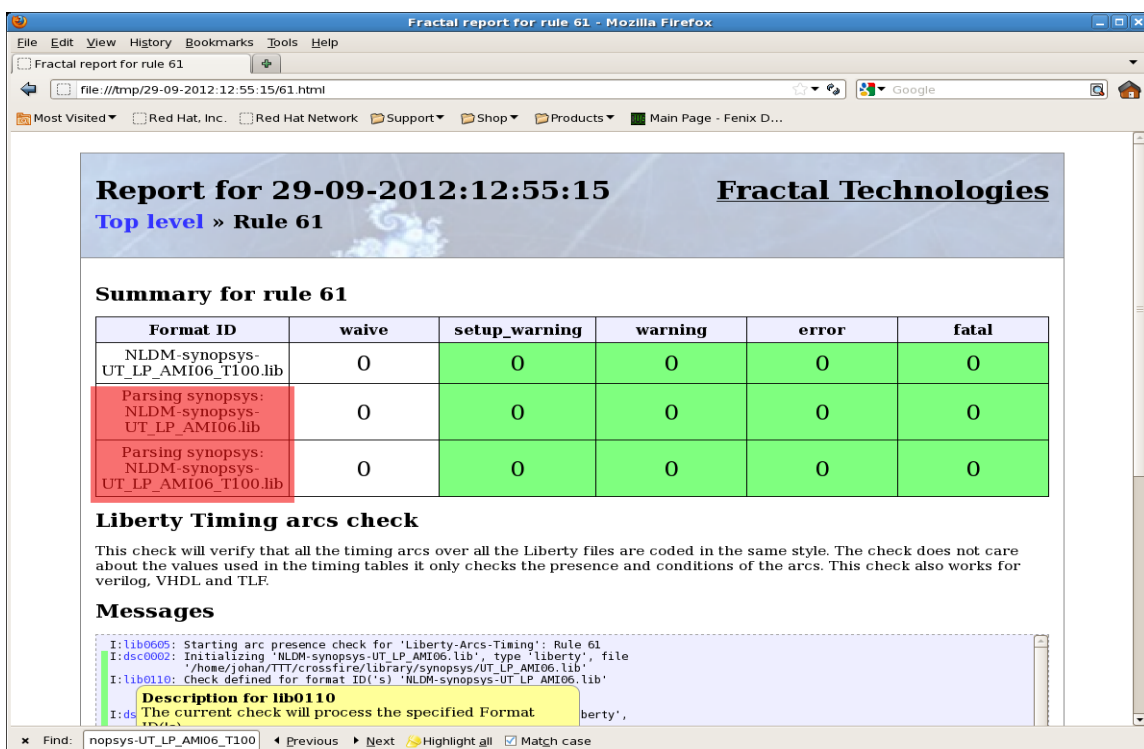
Summary

Rule number	Duration	waive	setup_warning	warning	error	fatal
61	1.731 sec	0	0	0	0	0
611,61	0.11 sec	0	0	1	0	0
611,61	0.11 sec	0	0	0	4	0
Total	2.783 sec	0	0	1	4	0

© 2012, Fractal Technologies Report generated on 2012-09-29, 12h:09m.

Find: nopsys-UT_LP_AMI06_T100 Previous Next Highlight all Match case

The report itself far transcends the notion of a simple summary of checks performed with a pass/fail status. For each check detailed descriptions of what is checked can be directly obtained from the report itself. Likewise for checks that don't pass detailed view-comparisons or other explanations serve to demonstrate the issue on the IP-data itself. Unfamiliar as this may seem to QA-outsiders, a final QA report may indeed contain failed checks. Crossfire offers the functionality of waiving checks that fail, provided that the designer offers a motivation for this particular waive. This allows an IP blocked to pass the entire mandatory check set, while at the same time offering to the IP-customer all the necessary insight into which checks were waived and why.



Report for 29-09-2012:12:55:15 **Fractal Technologies**
Top level » Rule 61

Summary for rule 61

Format ID	waive	setup_warning	warning	error	fatal
NLDM-synopsys-UT_LP_AMI06_T100.lib	0	0	0	0	0
Parsing synopsys: NLDM-synopsys-UT_LP_AMI06.lib	0	0	0	0	0
Parsing synopsys: NLDM-synopsys-UT_LP_AMI06_T100.lib	0	0	0	0	0

Liberty Timing arcs check

This check will verify that all the timing arcs over all the Liberty files are coded in the same style. The check does not care about the values used in the timing tables it only checks the presence and conditions of the arcs. This check also works for verilog, VHDL and TLF.

Messages

```
I:lib0605: Starting arc presence check for 'Liberty-Arcs-Timing': Rule 61
I:dsc0002: Initializing 'NLDM-synopsys-UT_LP_AMI06.lib', type 'liberty', file
'/home/johan/TTT/crossfire/library/synopsys/UT_LP_AMI06.lib'
I:lib0110: Check defined for format ID('s) 'NLDM-synopsys-UT_LP_AMI06.lib'
Description for lib0110
I:ds The current check will process the specified Format
berly',
```

Find: nopsys-UT_LP_AMI06_T100 Previous Next Highlight all Match case



QA Aspects of SoC Integration – Trust in God, but check your IP

The dominant QA aspect of SoC integration is validating the IP that is received from the various suppliers and their compatibility. Even though IP suppliers will provide proof-of-concepts with the data they deliver, SoC integrators will always need to perform incoming inspection on the libraries and IP they receive before plugging them into their design flow. The motivation is always to reduce the time-to-discovery for bugs found in the incoming models. Anything found during incoming inspection can be directly reported to the supplier and does not have the opportunity to disrupt the IP design at later stages, closer to the tape out deadline.

Crossfire is the enabling tool for quickly reaching a decision on whether a new IP or library shipment can be accepted. Once provided with the root-folder of the IP delivery, Crossfire will automatically recognize the delivered databases and formats, recognize the IP classification and select the appropriate checks that apply to these formats. A click-of-a-button later and the IP-receiver are presented with an elaborate incoming inspection report. As this report is specific to the IP delivered, evaluating the contents (in essence going over the failed tests and deciding to waive or fix them) is a straightforward process. Where needed the Crossfire API may be used to extend the built-in Crossfire checks with checks particular to the design-style deployed during SoC integration – think of naming conventions for different clock-domains.

The typical next step in an IP-supplier relationship is to move the qualification responsibility to the supplier. The IP or library supplier will take on the responsibility of proving that the delivered IP is indeed conforming to the standards agreed with the IP-consumer. Crossfire provides the ideal framework for such an engagement as the check-set for the IP qualification can be created by the IP-consumer and passed on to the IP-supplier. The incoming inspection work is then reduced to reviewing the Crossfire IP qualification report that was shipped with the IP delivery.

Such an arrangement allows for very rapid adaptation of new or modified IP blocks into the SoC design flow, making the entire SoC design process more predictable and manageable.

Conclusions

We have presented the QA requirements present in today's SoC design flow, which are made particularly challenging by large distributed design teams and the need to model deep submicron physical effects. It has been shown how the Fractal Crossfire product can serve as the QA backbone for an integrated approach to Quality Assurance. Crossfire can assist designers during their daily IP creation work and implements sign-off checks and incoming inspection checks surrounding the transfer of IP. Crossfire-check sets allow to substantially reducing the SoC integration time by transferring the qualification responsibility to IP suppliers through IP inspection check sets.

Thus Crossfire enables a truly Quality Assured SoC design flow across multiple partners, allowing predictable completion of SoC design projects.