Use-cases for Crossfire Error Fingerprint Visualization

A Fractal whitepaper

Introduction

Crossfire is well established as the industry standard IP qualification tool and as such has a key role both during IP design and hand-off of IP between designers and integrators. This achievement relies on the rigorous inspection of large IP databases for a wide range of industry standard quality checks regarding database consistency and timing and power characterization integrity. A recent addition to the Crossfire IP qualification tool is the visualization of the structure behind the errors that were found, known as the error fingerprint. This paper describes various use-cases for improving IP quality based on the insights gained from error fingerprint visualization.

A Crossfire error fingerprint is represented as a network with nodes representing the databases, cells, arcs and values that all belong to the same class of errors. An edge in the graph is drawn if both objects are associated with the same error. The following screenshot serves to further explain the concept:

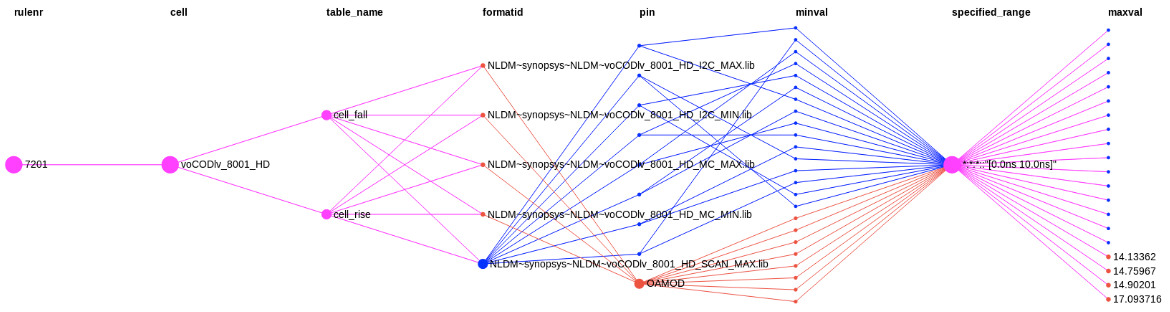

rulenr
cell
table_name
formatid
pin
minval
specified_range
maxval
7201
voCODlv_8001_HD
cell_fall
cell_rise
NLDM~synopsys~NLDM~voCODlv_8001_HD_I2C_MAX.lib
NLDM~synopsys~NLDM~voCODlv_8001_HD_I2C_MIN.lib
NLDM~synopsys~NLDM~voCODlv_8001_HD_MC_MAX.lib
NLDM~synopsys~NLDM~voCODlv_8001_HD_MC_MIN.lib
NLDM~synopsys~NLDM~voCODlv_8001_HD_SCAN_MAX.lib
OAMOD
::*:"[0.0ns 10.0ns]"
14.13362
14.75967
14.90201
17.093716

Figure 1, Crossfire error fingerprint example - What's wrong with the delays for the OAMOD pin?

The error visualized here is that for rule 7201: "Range check for cell_rise/cell_fall delay values". This check flags unrealistic values entered for cell-delays in Liberty files originating from characterization issues. What is deemed 'unrealistic' is of course up to the user to decide and will highly depend on the process node and the type of IP that is modeled.

In this example a default range of 0-10ns is considered realistic, anything outside that range is flagged. This is a typical way to start investigating characterization quality. One first sets a reasonable initial control limit, the corresponding error fingerprint then reveals the actual value-ranges that occur in the IP as well as the outliers that require further investigation. For this particular IP a waive up to 14ns maximum delay was also installed. A waive in effect still generates the same amount of errors but those error instances that satisfy the waive limits (rise/fall delays less than 14 ns in this case) will not contribute to the total reported error count. At a later point in time, when the IP is close to final release, this waive may expire: this would require either that all cells satisfy the required 10ns max-value or that the check-range is extended from 10 to 14ns.

Figure 1 represents all relevant instances of these range-check errors found in a particular Crossfire run. The first thing that can be noticed is that all of the 20 errors are generated for a single cell: the second column (“cells”) in the graph shows only 1 node. This particular cell has extremely high rise/fall delays for 5 different process corners (“formatid” column) and 9 different terminals (“pin” column). Hence it can be concluded that something systematic related to this individual cell is causing these extreme delay values.

The colors of the edges in the graph indicate the waiving status: blue represents waived errors, red are remaining violations and finally purple is a mix of the two. The red edges represent the errors that contribute to the total number of errors reported by Crossfire. From the error fingerprint it can be seen that all these remaining violations are associated with a single terminal: the OAMOD pin. Hence the conclusion that can be instantly read from the error fingerprint is this: the OAMOD pin delay characterization requires further investigation.

Reaching this conclusion by looking at the error fingerprint is so clear and obvious that it’s easy to take it for granted. Consider for a moment having to reach this same conclusion from just a list of error messages. If the waives are excluded only 5 errors are left, but recognizing that it’s always the same pin from the same cell is not straightforward. Double clicking an error in the Crossfire Diagnose environment takes the user straight to the relevant cell_rise- or cell_fall-table in the Liberty file. Although this is a great help for diving into the details of a particular error, it obfuscates the systematic aspects of this class of errors (same cell, same pin) which is so well illustrated by the error fingerprint visualization.

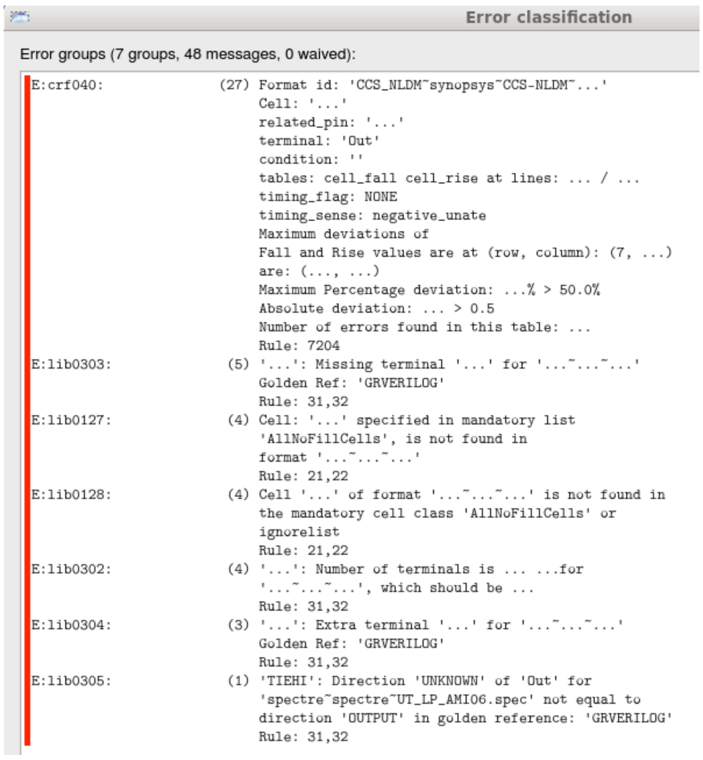
This example illustrates how the Crossfire error fingerprint reveals the common aspects of a class of related errors to guide users towards their final objective: to have clean IP. Error fingerprints show which formats, cells or tables need to be investigated for improvements to reach a higher level of IP quality.

Use cases

In this section several different usage scenarios will be described to illustrate the benefits of error fingerprint visualization. As shall be shown these can be summarized as time-savings and additional insights into the IP quality and structure.

1. **How to make sense of the errors?**

   In this scenario an IP developer has made an update to the design or characterization flow and runs the standard Crossfire checks before committing the changes for release to the integration team. Unexpectedly, Crossfire detects a 100’s of errors so as things stand these design changes cannot be accepted. Error fingerprint visualization allows the designer to quickly mine the error database to isolate the possible root causes. First Crossfire Diagnose will classify the errors, and provide the user with a pareto listing of the different error classes, see the figure below.



Error classification

Error groups (7 groups, 48 messages, 0 waived):

```
E:crf040:                (27) Format id: 'CCS_NLDM~synopsys~CCS-NLDM~...'
                              Cell: '...'
                              related_pin: '...'
                              terminal: 'Out'
                              condition: ''
                              tables: cell_fall cell_rise at lines: ... / ...
                              timing_flag: NONE
                              timing_sense: negative_unate
                              Maximum deviations of
                              Fall and Rise values are at (row, column): (7, ...)
                              are: (..., ...)
                              Maximum Percentage deviation: ...% > 50.0%
                              Absolute deviation: ... > 0.5
                              Number of errors found in this table: ...
                              Rule: 7204
E:lib0303:                (5) '...': Missing terminal '...' for '...~...~...'
                              Golden Ref: 'GRVERILOG'
                              Rule: 31,32
E:lib0127:                (4) Cell: '...' specified in mandatory list
                              'AllNoFillCells', is not found in
                              format '...~...~...'
                              Rule: 21,22
E:lib0128:                (4) Cell '...' of format '...~...~...' is not found in
                              the mandatory cell class 'AllNoFillCells' or
                              ignorelist
                              Rule: 21,22
E:lib0302:                (4) '...': Number of terminals is ... ...for
                              '...~...~...', which should be ...
                              Rule: 31,32
E:lib0304:                (3) '...': Extra terminal '...' for '...~...~...'
                              Golden Ref: 'GRVERILOG'
                              Rule: 31,32
E:lib0305:                (1) 'TIEHI': Direction 'UNKNOWN' of 'Out' for
                              'spectre~spectre~UT_LP_AMI06.spec' not equal to
                              direction 'OUTPUT' in golden reference: 'GRVERILOG'
                              Rule: 31,32
```

Figure 1, Crossfire error classification

From this listing the fingerprint of the highest individual error counts can be viewed as a graph thereby identifying the root-causes (specific cells/pins, Liberty constructs or control limits).

2. **If I fix this, what other errors are also resolved? What are the right control limits?**

   The nodes in the error fingerprint graphs also indicate the amount of errors represented by them, simply by displaying them as larger or smaller dots. See again figure 1 for an example: the single cell has a large dot as it represents all the errors. From the fingerprint users can directly decide what the effect of a particular fix would be. All relevant metrics, like a maximum delay or delta values between process corners, are directly visible in the fingerprint (see figure 3) so users can decide on the appropriate value for a waive or for a rule parameter change to cover all relevant values without being too tolerant.

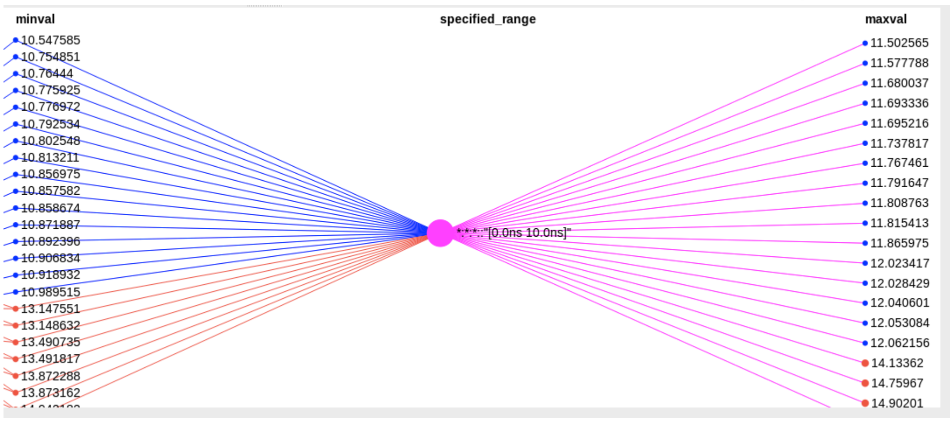


Figure 2, Rule metrics shown in error fingerprint

Note the difference with the traditional way of working using individual message inspection. Crossfire will highlight for every error instance the relevant values and locations in the IP database, but it would require flipping through all related errors to determine a sensible waive limit value. Using error fingerprint visualization the right value for a control limit can be directly read from the graph.

3. **Are these waives valid?**

Users receiving an IP shipment together with a Crossfire report will want to make sure that Crossfire was not tricked into passing the acceptance tests by simply waiving all the issues detected by Crossfire. As errors that are waived remain to be errors, users can still run an error classification using Crossfire Diagnose, pick out the most critical classes of errors and check the values in the error fingerprint. As all the waived values are directly shown in the graph, it immediately becomes clear where certain waives apply and to which cells/process-corners or timing/power modeling constructs.

To get the same information simply from investigating the waive descriptions is a lot less informative. Although the Crossfire rules and limits are clearly mentioned, the waive description does not mention the cells that are affected by the waive. This makes it difficult to decide whether a waive is relevant based on the waive description alone. Crossfire error fingerprints are able to support a correct judgement of the waive instantly.

4. **Is this error systematic or incidental?**

The error classification from Crossfire Diagnose guides users directly to the quality hotspots in an IP release. Especially when dealing with a new IP block, a new supplier or a new process node, error fingerprint visualization can help to decide whether a certain issue is systematic or incidental. Systematic issues will be repeated over e.g. all process corners or all cells of a particular category. The error fingerprint will feature an error-attractor with a fork-out of edges towards many different objects, all of which will be represented as small dots. As an example see the pin Q in the middle figure 4, which is missing from all databases.

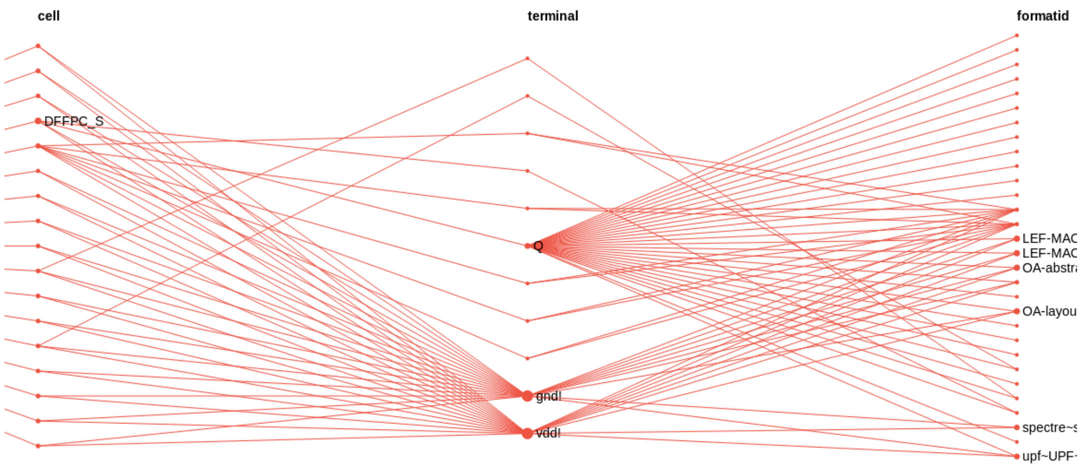


Figure 04, Systematic vs incidental fingerprint

An incidental issue on the other hand will show no repetition: figure 4 also shows a few incidental pins missing from a single cell in a single database. These occur only once and lead to fingerprints that consist of single paths through the fingerprint graph without any objects that attractor fork-out. These represent incidental issues that very likely each need individual fixes to have them resolved. The errors indicated for the power terminals illustrate an interesting blend: they are systematically missing from only a few incidental databases.

Conclusions

Crossfire error fingerprint visualization offers a next level in QA tool usability. Finding the errors is the first and foremost task of any QA inspection tool, but the final objective is to reach the next quality level by resolving the QA issues before the IP is integrated. By displaying a graphical fingerprint of every class of errors, Crossfire appeals directly to the highly developed image processing part of the human brain. This allows Crossfire users to reach conclusions quickly and provides additional insight in the structure and quality metrics of a particular IP release. At the end of the day error fingerprint visualization provides a faster path to IP quality improvement and allows Crossfire users to be more confident when they approve an IP release for QA acceptance.