The Evolution of Triage - Real-time Improvements in Debug Productivity

Gordon Allan, Mentor Graphics Corp.

1980s –
Same Engineers
do Design+Verif

1990s –
Designers Verify
Each Others’ RTL

2000s –
Separate
D/V/SW Teams With
Common Debug

2010s –
Triage Step
Avoids Duplicate Effort
Between Teams

10000 tests
1000 fails
100 symptoms
10 root causes

Triage:
microelectronic design/verification: the process of analysis of a discrete set of reported issues of common derivation, including a determination of priority and methodology for further exploration, analysis and ultimately resolution, of each issue, with the ultimate goal of making the most effective utilization of resources to achieve maximum benefit.

Several techniques can be used here to look at a fail in isolation and make observations about it which will help with other collective failure analysis activity. The main benefit of this initial preparation in triage is to know which resource or expertise is required and who should dig deeper on this particular fail.

1. Categorize an individual regression test failure
2. Analyze a list of N fails, look for commonality across multiple fails
3. Analyze list of NN fails looking deeper for common root causes across multiple fails
4. Analyze a single test instance across N regression runs of run history

Failure Mode

Failing Check Type

Symptoms/Hint in Log/Results

1. Categorize a single regression fail:
   - build-time error or a run-time fail condition or postprocessing?
   - testbench check or timeout or lack-of-success or mismatch?

2. Look for root cause indicators:
   - find the first failing error message in log, earliest assertion fired
   - are there uncharacteristic warnings, early indicators of failure?

3. Source and significance of the check:
   - designer inserted assertion or verification environment assertion?
   - scoreboard check or a protocol check - how precise is diagnosis?

4. What correlation can we see from the point of failure?
   - does same error appear in multiple failing test cases or configs?
   - can we correlate more cases after filtering out variant data?

5. Can we add precision to enable tool-based triage analysis?
   - categorize assertion behavior across multiple tests
   - locate precise signal path and time for further analysis, correlation

6. If less precision, can we identify interesting commonality?
   - which general area of design function or problem signal group?
   - anything in common in multiple failing tests to bin them together?

7. Use available formal technology:
   - analyze design around the point of failure
   - automatically apply formal property checking to identified region
   - spot common design rule violations

8. Use available debug root cause analysis tools:
   - run a causality check in the debug tool
   - two pieces of info: signal name, time of failing transition
   - derive root cause signal name and time, for further analysis

9. Repeat results analysis with these ‘enhanced’ results:
   - identify common signatures across regression run
   - correlate several different symptoms to one root cause

10. Has this failure mode occurred before?
    - what was this observation was last made
    - what was deduced about the problem on that occasion?

11. If this a noisy test?
    - does this test fail continuously/regularly or sporadically?
    - a new regression in behavior, or is testbench hitting fragility limit?

12. What can we learn from debug history and records kept?
    - when was this observation was last made
    - do we already know the likely root cause of this test fail or check?
    - who should we talk to, as allocate further debug?