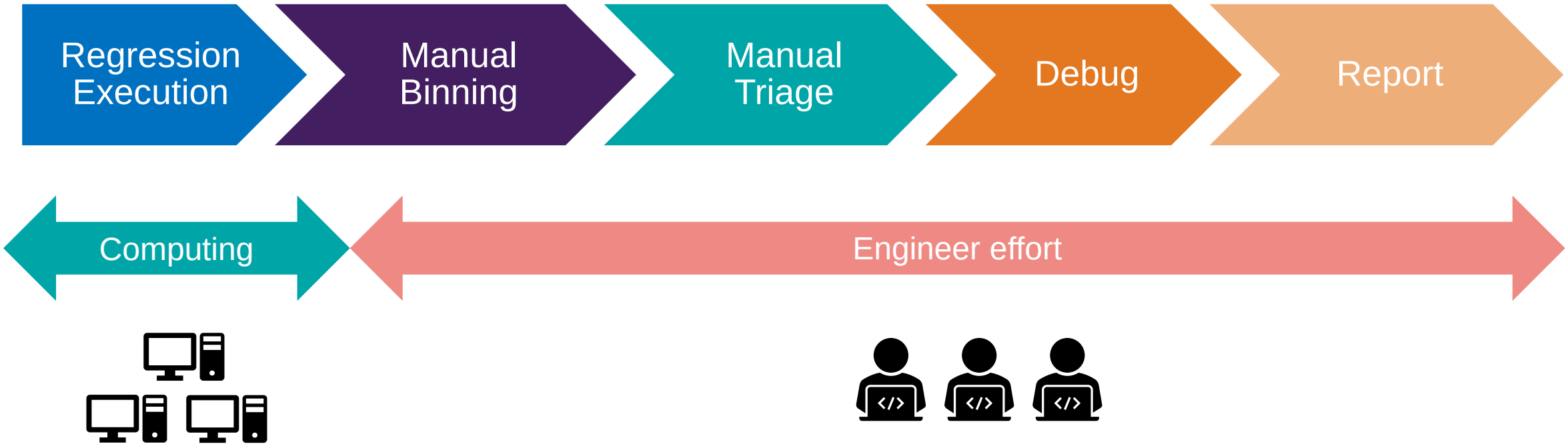
MEDIATEK
Connectivity Chip Design

snug

Verdi RDA:
A flexible regression debug accelerator

ChienLin Huang, MediaTek
Sherwin Lai, Stan Huang, Kai Yang, Alvin Chen, Synopsys

Traditional Regression Flow

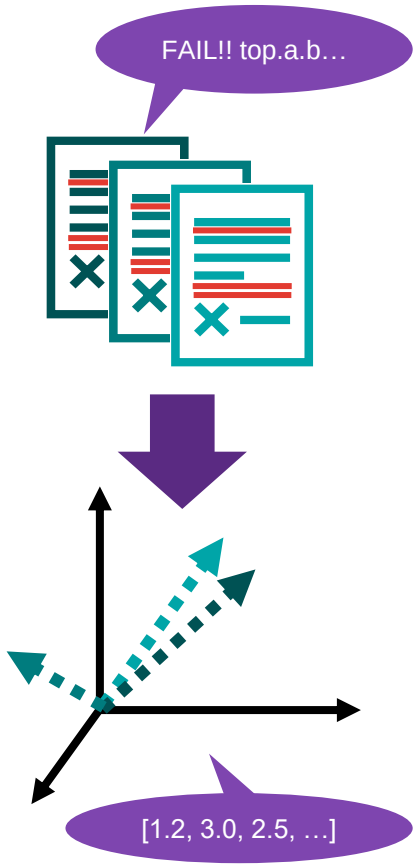


Pain points 1: Analyze and Classify Issues

ML based auto binning feature: Error binning

- Extract all error messages in simulation logs
  - Customized error messages acceptable
- Calculate the error shapes of failed simulations
  - Convert error messages to **error shapes**
- Conduct binning Into buckets
  - Group similar failures into the same bucket
  - Separate irrelevant failures into different buckets

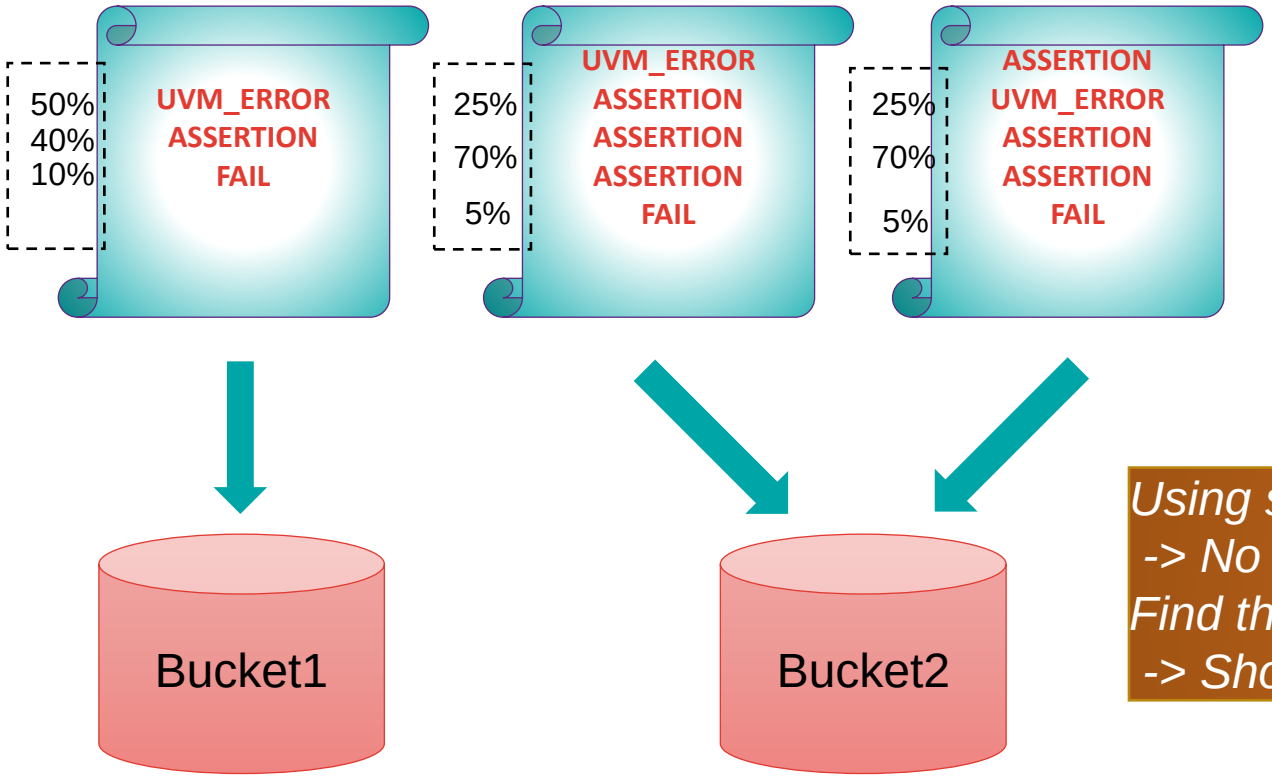
Classify the result with the press of a single button by anyone.
-> Save 4 hours 1 man per regression result
-> Save 8 hours 1 man for result parser



Pain points 1: Analyze and Classify Issues

ML based auto binning feature: Multiple error binning + user defined error

- Turn on with **binning.multiple_error_binning: True** in config yaml
- Grouping by the distribution of all the error messages in each logs

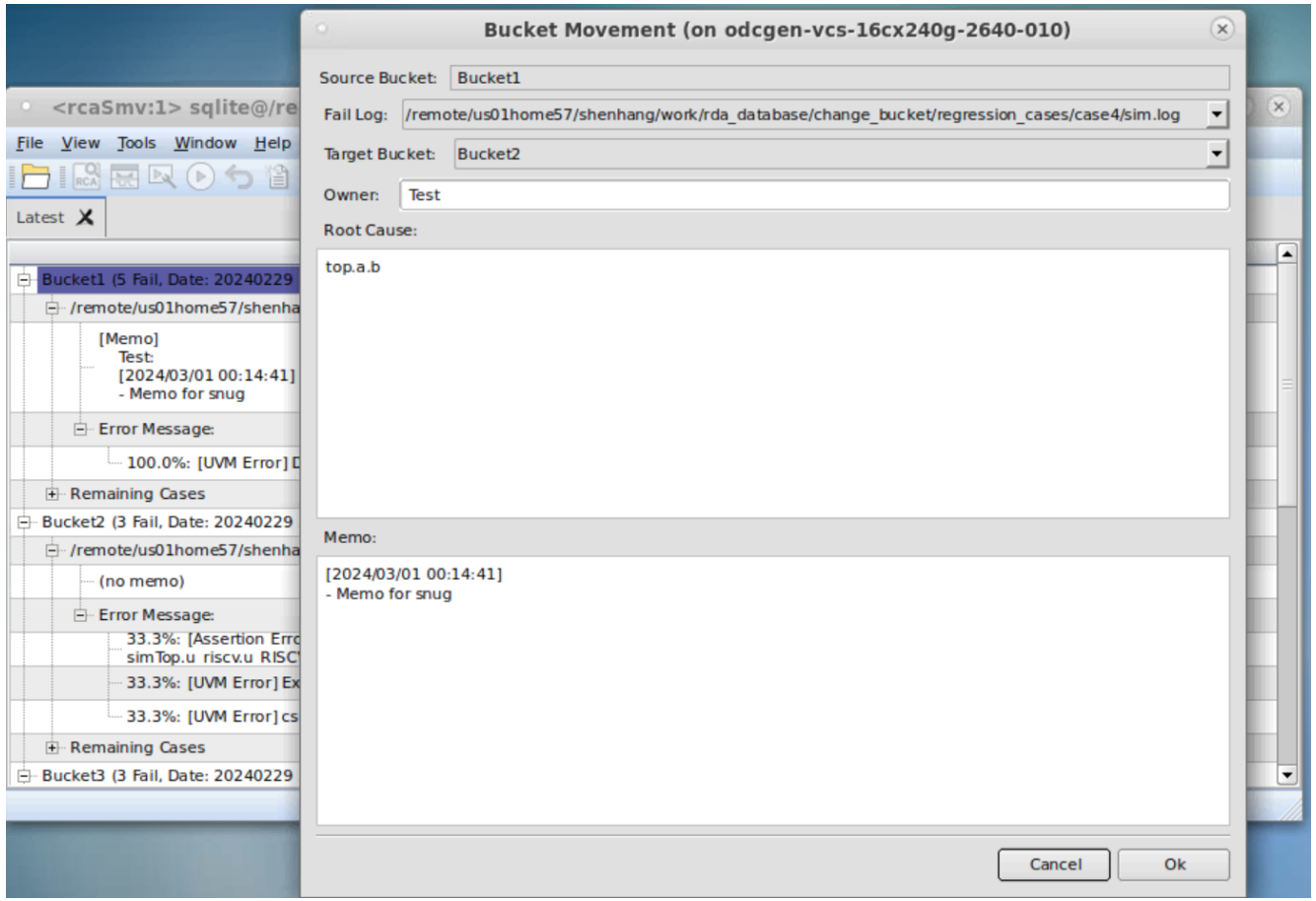


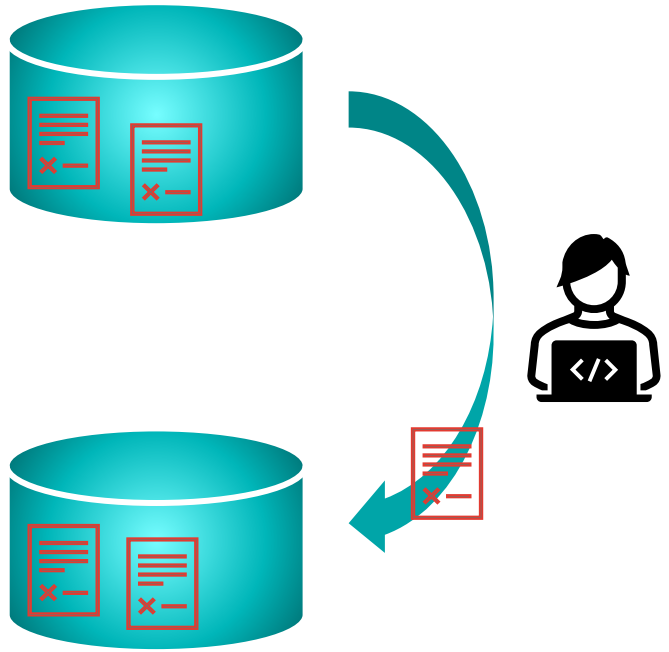

```
binning:
  multiple_error_binning: True
  user_defined_rule: ./binning_rule.txt
```

Using simple regular expression
-> No need rewrite checker for RDA
Find the multiple design bugs at one shot
-> Shorten the verification cycle by approximately 20%.

Pain points 2: Assign and tracking debug status

Memo system in the auto binning feature: memo + bucket movement





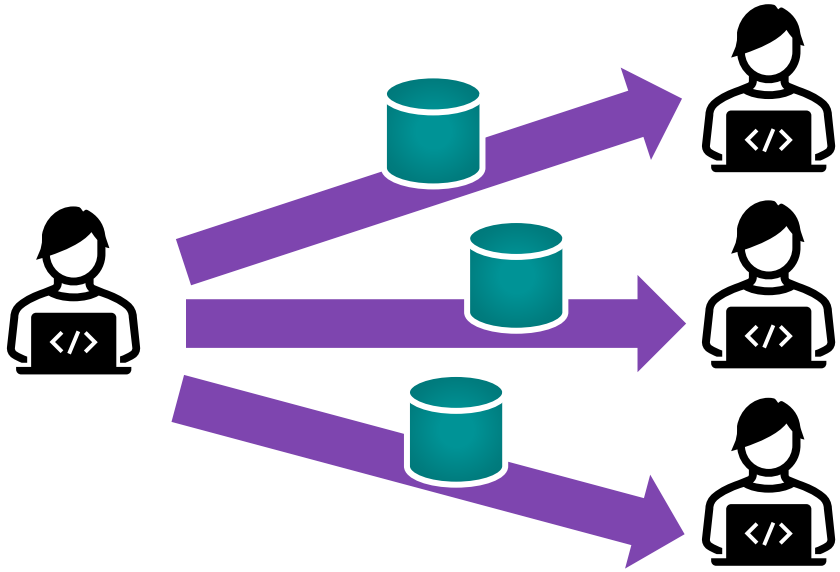
Use the memo system instead of email.
-> more straightforward
-> no more mail bomb

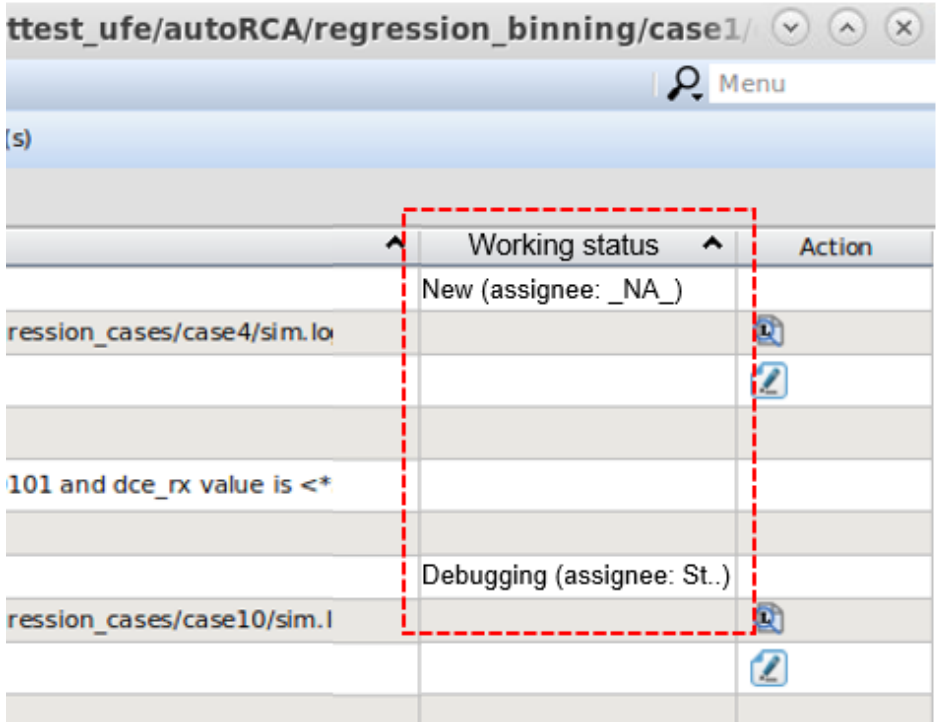
- Adjust binning results based on users' knowledge flexibly

Pain points 2: Assign and tracking debug status

Memo system in the auto binning feature: assignment collaboration

- Collaboration with RDA instead of mailing
  - Assign buckets to the right assignee
  - Leave a memo
  - Review debugging status



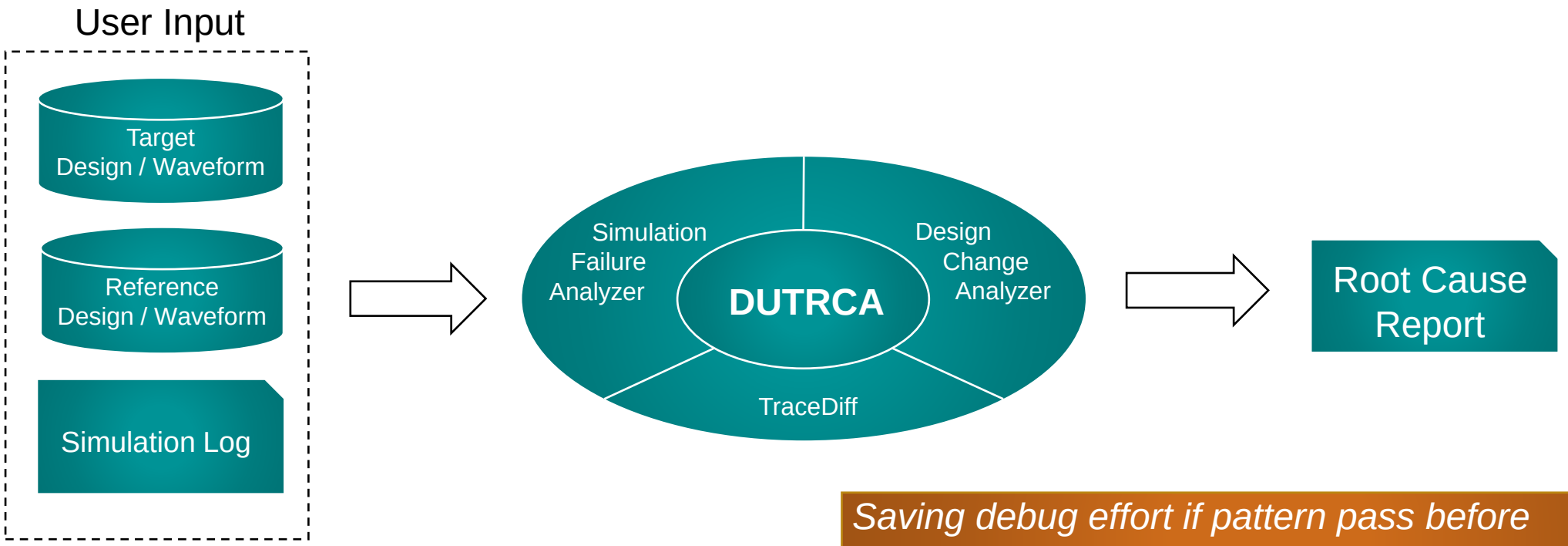


Analyze -> assign -> report in one tool
-> no more issue tracking system, save lots of $$
-> reduce 20% effort for issue assign and tracking

Pain points 3: Testcase pass before

DUTRCA / TBRCA feature

- Apply technology of “Log Analyzer”, “TraceDiff” and “Design Change Detection” to find suspected fail case root cause automatically

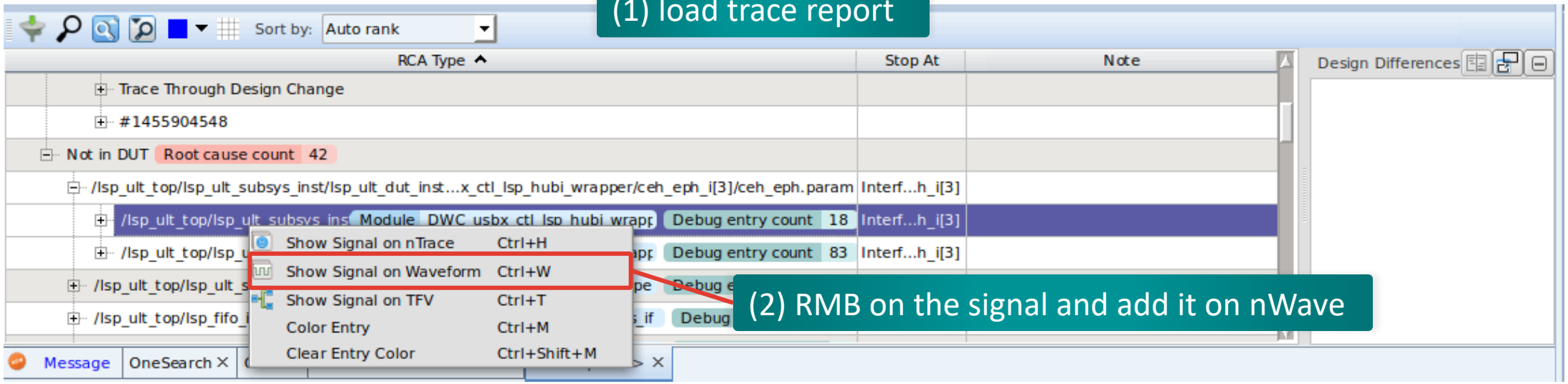


Saving debug effort if pattern pass before
-> save 90% effort for debug fail pattern

Pain points 3: Testcase pass before

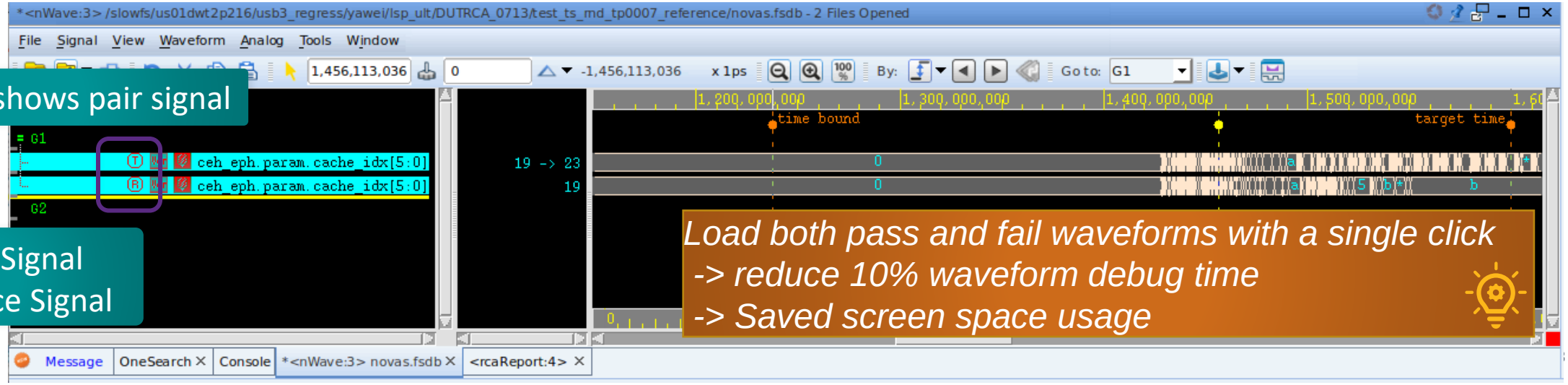
DUTRCA / TBRCA feature: Add pair signals on Waveform

(1) load trace report



(2) RMB on the signal and add it on nWave

(3) nWave shows pair signal

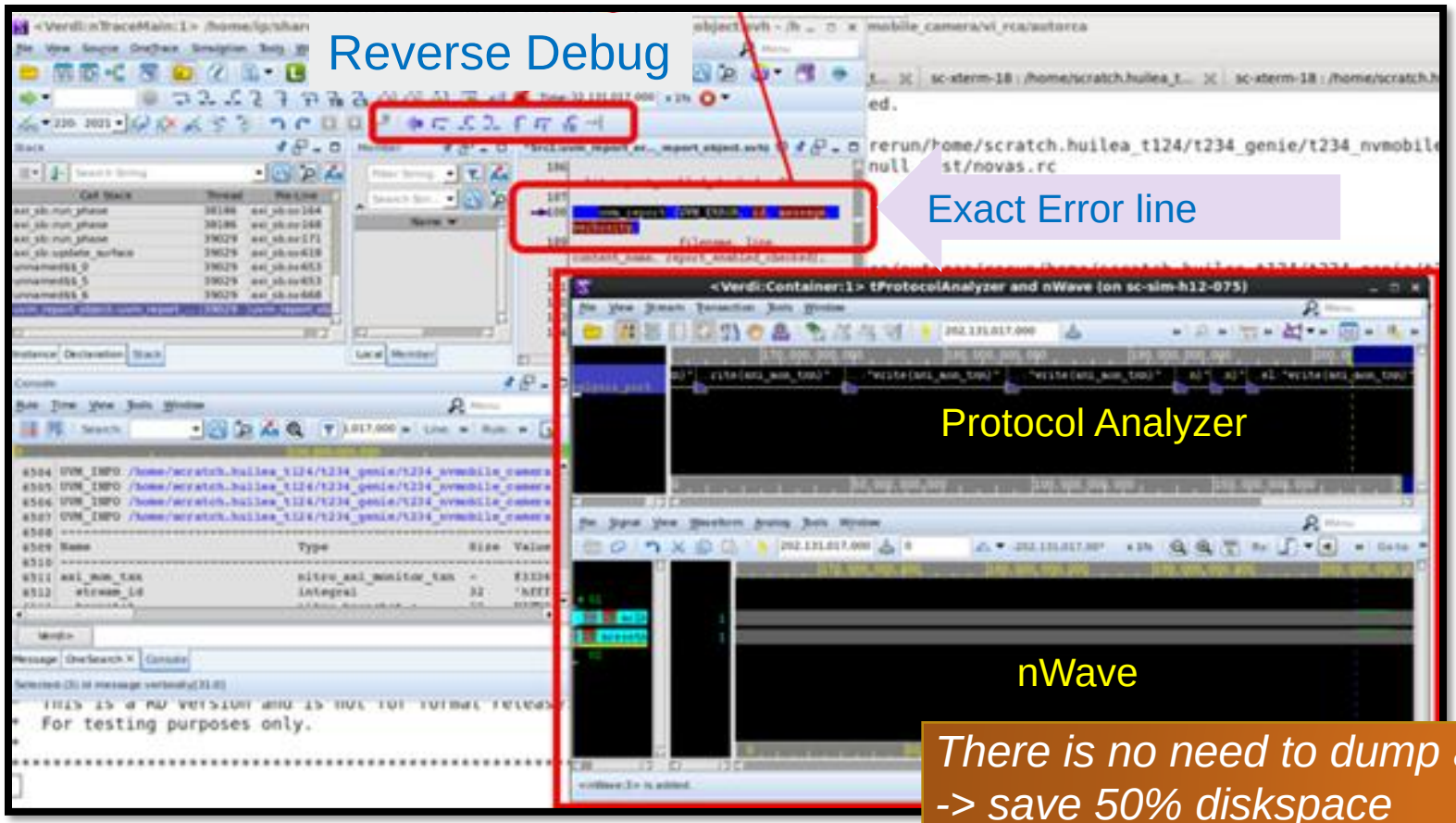


T: Target Signal
R: Reference Signal

Load both pass and fail waveforms with a single click
-> reduce 10% waveform debug time
-> Saved screen space usage

Pain points 4: Rerun for dump waveform

Debug facilitator feature: Interactive reverse debug



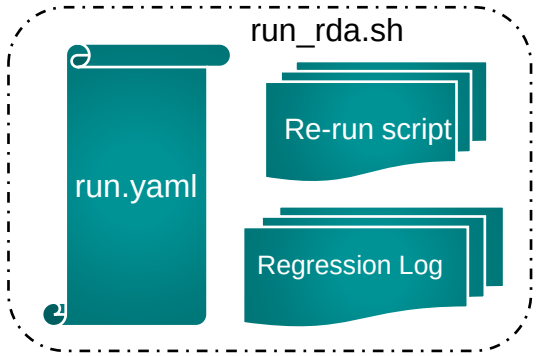
- Automatically stop at the error point
- Stacked view of synchronize protocol analyzer & nWave
- Message Analyzer Report

There is no need to dump all failed waveforms before debugging
-> save 50% diskspace
-> reduce 10% simulation time for waveform dumping

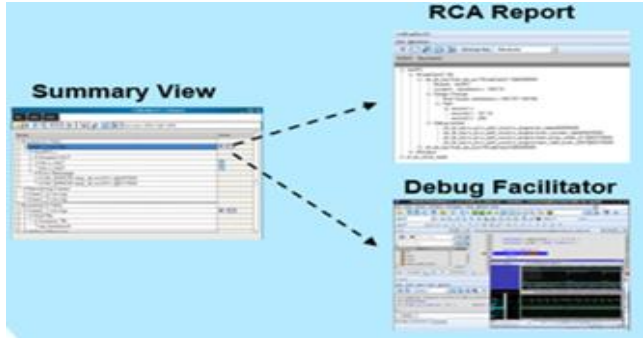
Binning + DUTRCA + DF

= Flexible Regression Debug Automation System

RDA input data



RDA Manager



Many stages can be customized -> It is easy to integrate into the current regression flow

Invoke RDA
autorca –cfg run.yaml

Check Report & Debug
autorca –load_report report.json

RDA DB

Regression Nightly Run

RDA Nightly Run

Run regression

Get Sim Log List

Customized

RDA – Auto-Binning

Rerun fail cases to generate debug data

Customized

Run DUTRCA

Customized

Conclusions

- RDA is a system that integrates log analysis, error binning, issue assignment and tracking, root cause analysis, and on-the-fly debugging within a single Verdi GUI. In our test cases, this system
  - Saved the effort of one person
  - Two in-house tools
  - Reduced designer debug effort by 50%
  - Shortened the cycle time for single regression issue debugging by 20%.

- The DUTRCA feature provides regression owners with a very powerful tool, saving 90% of the debug effort, although it is currently only applicable to fixed patterns.

- Since all tasks can be completed within the same interface, the feedback from team members has been very positive.

- The difficulty of implementation has been reduced by 50% due to the ability to customize in many places through options. Of course, some scripts will still be needed to interface with the original regression flow.

Future Works

Historical analysis with database

- Provide debugging recommendations and hints
  - Continuous failing bucket, new failing bucket…
- Detect similar bucket-fixed patterns before
  - Reduce redundant effort
- **Critical debugging, effective debugging**


THANK YOU
Our Technology, Your Innovation™