Debug Automation with AI

Craig Yang, Jaw Lee, Sherwin Lai
Verdi R&D, Synopsys Taiwan
Motivation and Debug Flow
**Regression Debug Automation Motivation**

- **Regression Execution**
  - Manual Binning
  - Manual Triage
  - Debug

- **Debug and Validation with RCA report**

- **Engineer effort**
  - Reduce engineering effort and TAT with ML and advanced RCA technologies

- **Computing**
  - Regression Execution
  - Automated Binning
  - Failure Triage and Analysis
  - Debug

- **Engineer Effort**
  - Computing
Regression Debug Flow with ML

Regression Run

Probe sim Re-run

Reference snapshot*

Regression Automation

Execution Manager

Debug GUI

RCA Manager

Debug Facilitator

RCA Report

Debug Engines

DUT RCA

XRCA

TB RCA

Log/Msg Analysis

Codes check-in

Failing tests

Passing tests

ML Binning

Clusters

ML Smart Probe

Execution Manager

Bucketize reg failures.
Representative case picked

Bug prediction, code change checked-in that cause failures

If ref. snapshot found, debug engines to root cause the bugs

Debug GUI to complement the debug engines.
RCA Manager and Report for engineer to review

User consumable results

Regression Run

Passing tests

Failing tests

Debug Engines

DUT RCA

XRCA

TB RCA

Log/Msg Analysis

Debug GUI

RCA Manager

Debug Facilitator

RCA Report

User consumable results

Regression Run

Regression Run

Regression Run

Regression Run

Regression Run

Regression Run

Regression Run
Regression Binning
Regression Binning with ML

- Apply machine learning technology for regression binning
  - Perform failure clustering based on the similarity of simulation logs
Useful Regression Binning Features

• Predefined Errors
  • UVM Errors, OVM Errors, SVA Failures, etc.

• User Defined Errors
  • Control the Message Chosen Mechanism
    • Filter Rules
    • Waive Rules
    • Rule Priority
  • Adjust the Binning Result
    • Replace Rule
    • Pass Rule

• Multiple-Error Binning
Bug Prediction
Probe Engine

- Some debug engines need a reference design snapshot
- We need a probe engine to find the reference design snapshot
Traditional Probe without Bug Prediction

- **Culprit Changelist**
  - Failure observed
  - Divide and conquer
  - Find an adjacent pass / fail pair
  - Resource Intensive!!

- **Culprit Changelist**
  - Rerun #1 (F)
  - Rerun #2 (P)
  - Rerun #3 (F)
  - Rerun #4 (P)
Bug Prediction

Failed CLs

Passed CLs

Collect metrics for failed and passed CLs

Bug Prediction Model

Predict suspicious CLs

Prioritize the probe efforts to the suspect changelists that are most probable to cause potential failures

Rerun to validate prediction

1. Passed CL observed last time

2. Failed CL observed this time

11
Bug Prediction for Smart Probe

- Prioritize the probe efforts to high-risk changelists
Bug Prediction for Risk Assessment

• Help the users assess high-risk changelists without running probe
Message Analysis
Message Analysis – High Level Flow

1. Simulation Log File
2. Collect messages to be analyzed
3. NLP
4. FSDB
5. Match transactions
6. Matching Signals
7. Collect Attributes
8. Collect Values
9. Pair Attribute-Values
10. Matching transactions
11. Matching Signals
12. Refine and Rank Transactions
13. Identify Related Interface Signals (VIF)
14. Transaction and Signal Viewer
15. Report
Message Analysis – Example

TBAR INFO: Started analyzing debug messages from simulation log file

TBAR INFO: Simulation log file contains EMU07344 debug messages.

TBAR INFO: Analyzing the first EMU07344 log message.

TBAR INFO: Analyzing the below log message.

TBAR INFO: Completed analyzing debug messages from simulation log file.

TBAR INFO: Matching attribute/valued are listed below.

TBAR INFO: Matching transactions after analyzing actual, next and previous messages are listed below.

TBAR INFO: Stream: ([trans rooted/ Tran test top/Ubuntu example/Ubuntu/Drivers/seq item port] Matching transactions: (2)

TBAR INFO: Stream: ([trans rooted/ Tran test top/Ubuntu example/Ubuntu/Drivers/seq item port] Matching transactions: (1)

TBAR INFO: Total matching transactions (5)

TBAR INFO: Matching virtual interface paths are listed below.

TBAR INFO: UTC* Alg signals: (seq read seq write seq addr)

TBAR INFO: Top ranked transaction details are listed below.

TBAR INFO: Connected to (Tran test top/Ubuntu example/Ubuntu/Drivers/seq item port) Component type (seq driver) Port type (seq item pull port)

TBAR INFO: Please run the (javelin)/vsip/ipl/TranAutoMACA/reports/simErrorAnalysis.log (run_verb) script to view the sim log file analysis results in Verdi.

Larger set of transactions

Smaller set of Related transactions

Interested transaction attributes

Matching transaction details

Simulation log message
Summary

• Manual regression debug is tedious, but we can automate it with AI and advanced RCA technologies

• **Regression binning** classifies many failed tests into a few bins of different errors

• **Bug prediction** reduces the time on locating reference snapshots for debug engines

• **Message analysis** identifies transactions and VIFs that are related to the error message
Introducing Next-Generation Verdi Platform

See the demo in the Synopsys booth!
Questions