Hardware Bug Triage Using Machine Learning

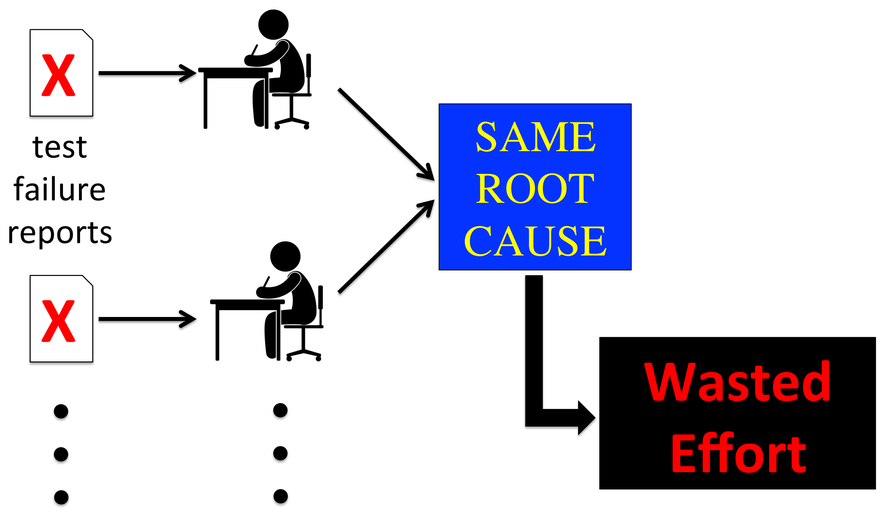
Rico Angell, Ben Oztalay, Noel Bhattacharyya and Andrew DeOrio

University of Michigan

DAC 2015

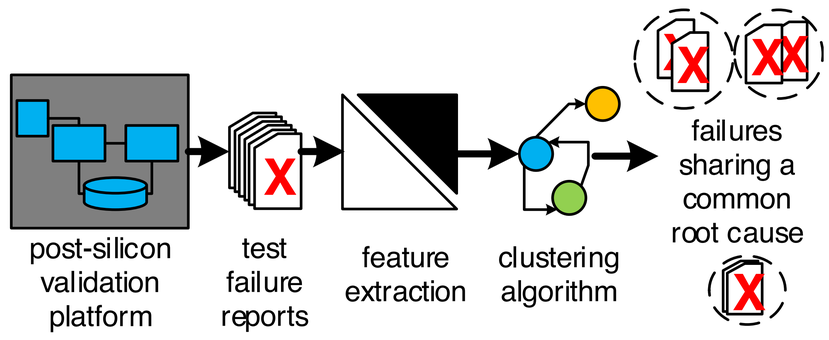
Problem

- Digital designs are continuing to become more complex and verification effort is increasingly burdened by post-silicon validation
- A bottleneck in the post-silicon validation process: Engineering resources are wasted debugging multiple test failures that are found to have been caused by the same root cause bug



Hardware Bug Triage Algorithm

- Start with a database of failure reports that include values of some of the control signals in the design
- Select features from log files in order to reduce each log file to a point in Euclidean space (module specific control wires)
- Reduce each failure report in the database to a point in Euclidean space based on the selection of features
- Use *k*-means clustering on data points
- The result is groups of failure reports such that the failures in a fixed group has the same root cause bug

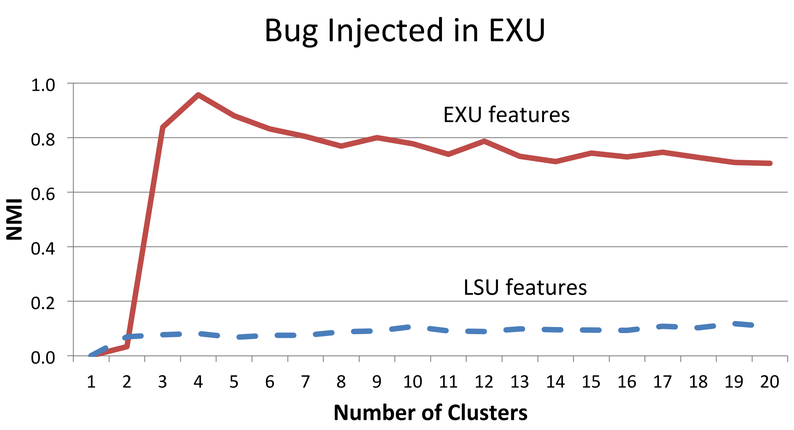


Module	Description
EXU	execution unit's register mgt. logic
DEC	decoder
TLU	trap logic unit
MMU	memory management unit
PMU	performance monitoring unit control logic
PKU	thread pick unit
FGU	floating point and graphics unit
GKT	gasket interface
LSU	Load/store unit
IFU	Instruction fetch unit

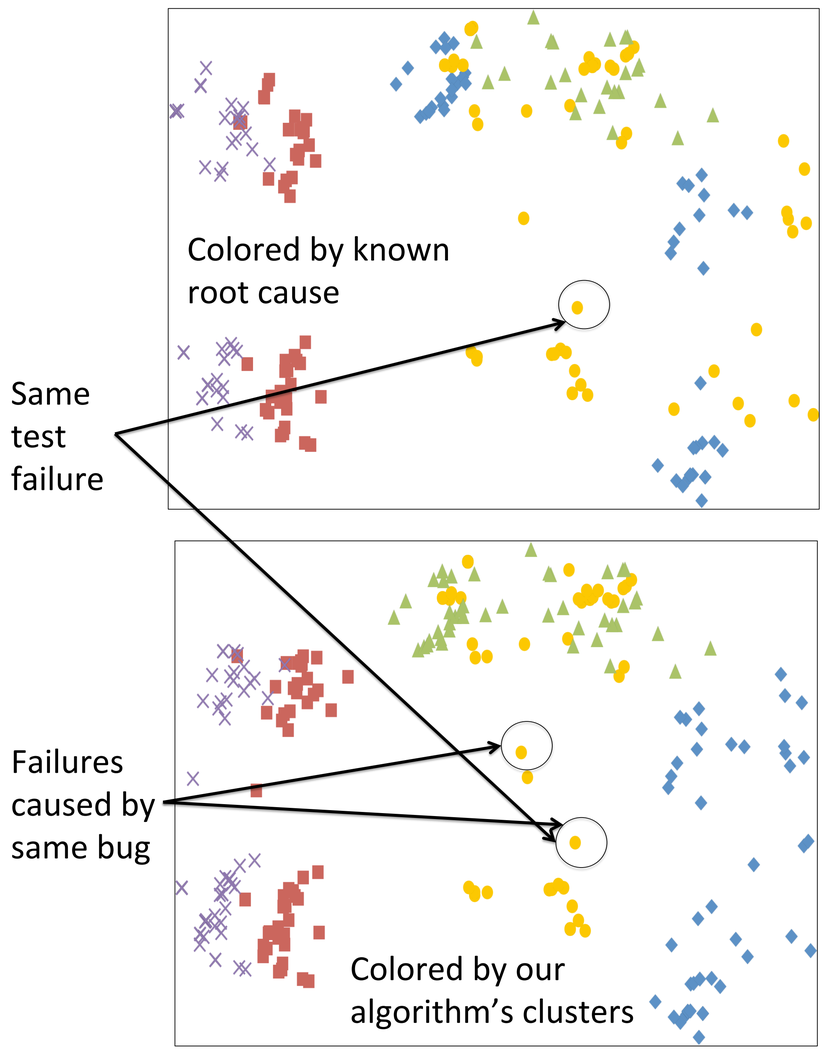
Modules injected with post-silicon bugs. Each module of the industrial-sized OpenSPARC T2 contained 5 stuck-at bugs.

Results

- Design bugs for submodules of OpenSPARC T2 using a hardware description language
- Run simulations that mimic the post-silicon validation testing process
- Each simulation dumps the values of all control signals in each module
- We use Normalized Mutaul Information (NMI) to analyze the quality of the clustering
- Higher NMI indicates more accurate clustering
- We accurately distilled 3,634 test failures into 50 groups



Effect of the number of clusters on quality of results using bugs injected in the execution unit (EXU) module.



Bug Injection location	Features used by clustering: EXU	PKU	DEC	MMU	TLU	PMU	IFU	GKT	FGU	LSU
EXU	0.88	0.09	0.13	0.07	0.06	0.04	0.07	0.03	0.01	0.07
PKU	0.00	0.33	0.03	0.00	0.00	0.00	0.01	0.01	0.01	0.01
DEC	0.23	0.17	0.40	0.20	0.21	0.24	0.22	0.16	0.14	0.21
MMU	0.43	0.54	0.44	0.83	0.48	0.43	0.48	0.39	0.38	0.43
TLU	0.48	0.45	0.41	0.44	0.72	0.48	0.43	0.47	0.36	0.43
PMU	0.45	0.49	0.44	0.47	0.46	0.69	0.43	0.44	0.38	0.44
IFU	0.07	0.17	0.13	0.10	0.13	0.11	0.31	0.01	0.02	0.11
GKT	0.40	0.38	0.36	0.41	0.40	0.37	0.46	0.80	0.32	0.36
FGU	0.32	0.42	0.38	0.31	0.31	0.32	0.31	0.29	0.31	0.31
LSU	0.50	0.58	0.50	0.52	0.49	0.57	0.52	0.46	0.47	0.50

Quality of clusters identified by our algorithm, expressed as NMI between our algorithm and a theoretical perfect clustering. The diagonal shows the modules where our algorithm correctly identified the root causes of the failures.

Conclusions

Our algorithm:

- Increases debugging efficiency by identifying test failures that share a common root cause
- Tolerates inconsistent failures and noisy post-silicon chips
- Automatically clusters failures with limited post-silicon signal visibility and without a golden model

Our results:

- Provide insights on applying machine learning to the dynamic signal activity in a complex digital design