Machine Learning-Guided Stimulus Generation for Functional Verification

S. Gogri, J. Hu, A. Tyagi, M. Quinn
S. Ramachandran, F. Batool, A. Jagadeesh

Texas A&M University
Outline

• Challenge of functional verification
• Background
• Previous work
• Machine learning guided stimulus generation
  • Coarse-grained test-level pruning and results
  • Fine-grained transaction-level optimization and results
• Conclusions
Simulation-Based Functional Verification

- Functional verification mostly via simulations
- Exercise stimulus and observe response
- Huge space
- Verification time and manpower > design

Push the curve toward left
Stimulus

Constraints applied at test-level are called test-knobs
Coverage

Coverage Metrics

Code Coverage
- Line/Block
- Toggle
- Condition

Functional Coverage
- Primary Target

Cover-Groups
- Cover-Points
- Bins (0/1)
Machine Learning for Fast Coverage

- A machine learning model
- Tells if a stimulus $\psi$ will cover an unverified point
- Simulate $\psi$ only if the answer is yes
Prior Art

- ML for functional verification was started 16 years ago
- Earlier than the recent booming of deep learning
- Mostly tried a model and then showed verification time reduction

Coverage Directed Test Generation for Functional Verification using Bayesian Networks

Shai Fine  
Avi Ziv  
IBM Research Laboratory in Haifa  
Haifa, 31905, Israel  
{fshai, aziv}@il.ibm.com
Some Previous Works

• Fine and Ziv, DAC 03, Bayesian network
• Guzeya, et al, TCAD 10, SVM
• Ioannides and Eder, TODAES 12, “Coverage-directed test generation automated by machine learning”
• Chen, Wang, Bhadra and Abadir, DAC 13, knowledge reuse
• Sokorac, DVCON17, genetic algorithm for toggle coverage
• Wang, et al, Great Lake Symp. VLSI 18, neural network
Are Existing Techniques Adequate?

- Mostly based on old ML engines
- No study on the granularity of ML application
  - Coarse-grained test level stimulus optimization
  - Fine-grained transaction level stimulus optimization
- Stimulus pruning? or constructive stimulus generation?
- No differentiation between Finite State Machine (FSM) and non-FSM design
Test-Level Stimulus Pruning

Phase I:
- Random test generation
- ML model is trained

Transition decided by online validation

Phase II:
- ML model is applied for test pruning
- ML model continues to be trained

Coverage Profile

Bug Rate

 Weeks

% Coverage

Failure Rate
ML Model Setup

• One ML model for each cover point

• For each model
  • One binary output for each cover bin
  • 1: the bin will be hit by a test
  • 0: the bin will not be hit by a test

• A test is simulated if it will hit any uncover bin
ML Model Features

Features
- Seed
- #transactions
- Core selection
- $type
- Request type
- …

Cover points
- Address X req type in bins
- Snoop request
- $protocol transitions
- $hit on each address
- …
Ternary Classification

- Will a test improve verification coverage?
- Conventionally: binary classification – yes or no
- Our approach:
  - Probability $p$ of improving coverage by test $\psi$
  - If $p$ is high, simulate $\psi$
  - If $p$ is low, do not simulate $\psi$
  - If $p$ is in middle, simulate $\psi$ and use the result to train ML model
Decision Tree Classification

Temperature Prediction Decision Tree

Season = Winter

- Historical average > 60
  - False
  - Temp Today > 55
    - True
    - Predictions: 58
  - False
    - Predictions: 62
- True
  - Temp Today > 65
    - False
    - Predictions: 71
  - True
  - Predictions: 40

- Historical average < 50
  - False
  - Temp Today > 45
    - False
    - Predictions: 54
  - True
    - Predictions: 22
- True
  - Temp Today > 25
    - False
    - Predictions: 40
Random Forest Classifier

[Diagram showing a Random Forest Classifier with three trees labeled Tree-1, Tree-2, and Tree-n, each leading to a class label (Class-A, Class-B). The final class is determined by majority voting.]
Test-Level Results: Group A

Covergroup A: coverage metrics correlate with test knobs

- Simulating selected tests
- Simulating all the tests

![Graph showing functional coverage vs. number of tests](image)

- No Learning
- DNN
- Random Forest
- SVM

![Bar chart showing # simulated tests](image)

- Ternary
- Binary

- No ML
- SVM
- Deep Neural Network
- Random Forest
Test-Level Results: Group B

Covergroup B: coverage metrics do not correlate with test knobs

Prediction starts here

100% Coverage

No Learning  DNN  Random Forest  SVM
Transaction-Level Stimulus Optimization

Finer-grained control than test-level pruning
Offline Sequence Generation for FSM

- Coverage metric: state transitions
- ML model: given current state and transaction attribute, predict the next state
- Phase 1: random simulation while ML model is trained
- Phase 2: generate transaction sequences leading to new transitions
Sequence Generation by Graph

- **Path-1**
  - Initial state: I_I_I_I
  - Transition: I_L_M_I → M_L_L_I
  - Terminate at illegal state: S_S_S_S

- **Path-2**
  - Initial state: I_L_L_L
  - Transition: I_L_S_S → S_L_L_S

**Sequence = longest path**

<table>
<thead>
<tr>
<th>Edge Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visited edge</td>
<td>-</td>
</tr>
<tr>
<td>Unvisited and predicted edge</td>
<td>-</td>
</tr>
<tr>
<td>Unvisited and illegal prediction</td>
<td>-</td>
</tr>
</tbody>
</table>
Online Pruning vs. Offline Sequence Generation

• Online transaction pruning
  • Myopic scope at each pruning

• Offline sequence generation
  • Much longer horizon in scope
FSM Transaction Optimization Results

Coverage Metric: MESI state transitions – 143 bins

Deep Neural Network (DNN) 48% reduction in simulation cycles
FSM Transaction Optimization Results

Coverage Metric: MESI state transitions – 143 bins

Random Forest Classifier (RF) 55% reduction in simulation cycles
FSM Verification Time

ML engine: random forest

Verification time reduction

57% Transaction Pruning
69% Directed Sequence

No Learning  Test-level Opt
Transaction Pruning  Directed Sequence
Non-FSM Event Coverage

• Events: buffer full, cache hit, etc.

• Almost impossible to deterministically cover events through test-level optimization

• Event coverage depends on transaction history
Recurrent Neural Network (RNN)

Unrolling over time, accounting for history
Long Short-Term Memory (LSTM)

\[ \sigma \]: gate function, allowing a signal to pass or not

LSTM applications: time series, natural language processing
History Effect

![Graph showing the History Effect](image)

- Error vs. Transaction Window Size (w)
- Comparison between Random Forest and LSTM Model
Non-FSM Event Coverage Results

Coverage Metric: cache hit on every address – 768 bins

Long Short-Term Memory (LSTM) 61% reduction in simulation cycles
Non-FSM Verification Time

Verification time reduction

65% Transaction Pruning
72% Directed Sequence
Conclusions

• Machine learning-based stimulus optimization for functional verification
• Fine-grained transaction level optimization outperforms coarse-grained test level pruning
• Offline sequence generation is superior to online stimulus pruning
• Random forest and LSTM are helpful
• Around 70% simulation time reduction
Future Research

• Small testcases
• Will work on big cases
• Colleagues with decades of industrial verification experience
• Seek industrial collaboration

Aakash Tyagi

Mike Quinn
Thank You!

Questions?