2025 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

MUNICH, GERMANY
OCTOBER 14-15, 2025

Accelerating Coverage Closure with Reinforcement Learning: A Case Study on FSM Verification

Tijana Misic, Verification Consulting

accellera SYSTEMS INITIATIVE

Agenda

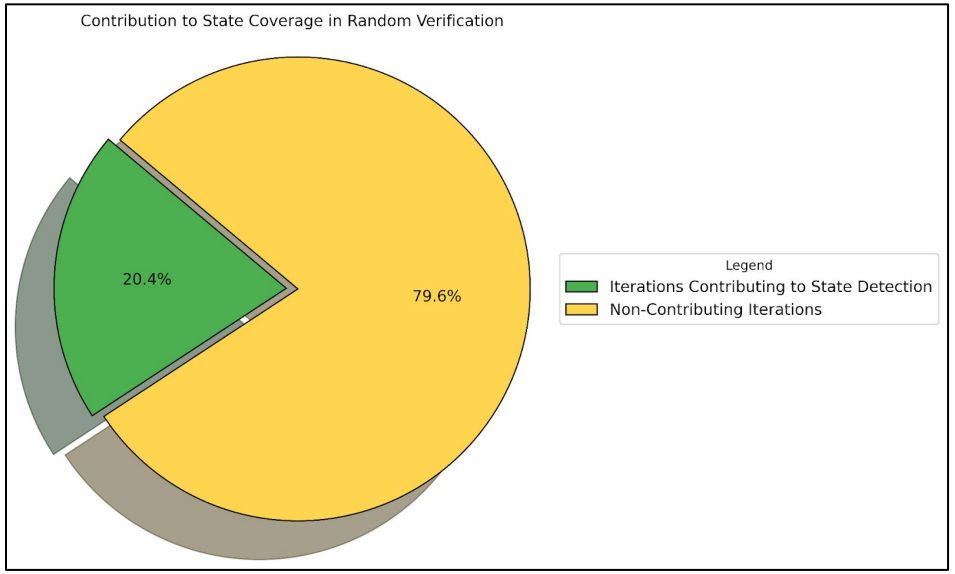
- Motivation
- Proposed Approach
- Implementation Results
- ROI & Discussion
- Conclusion

Motivation

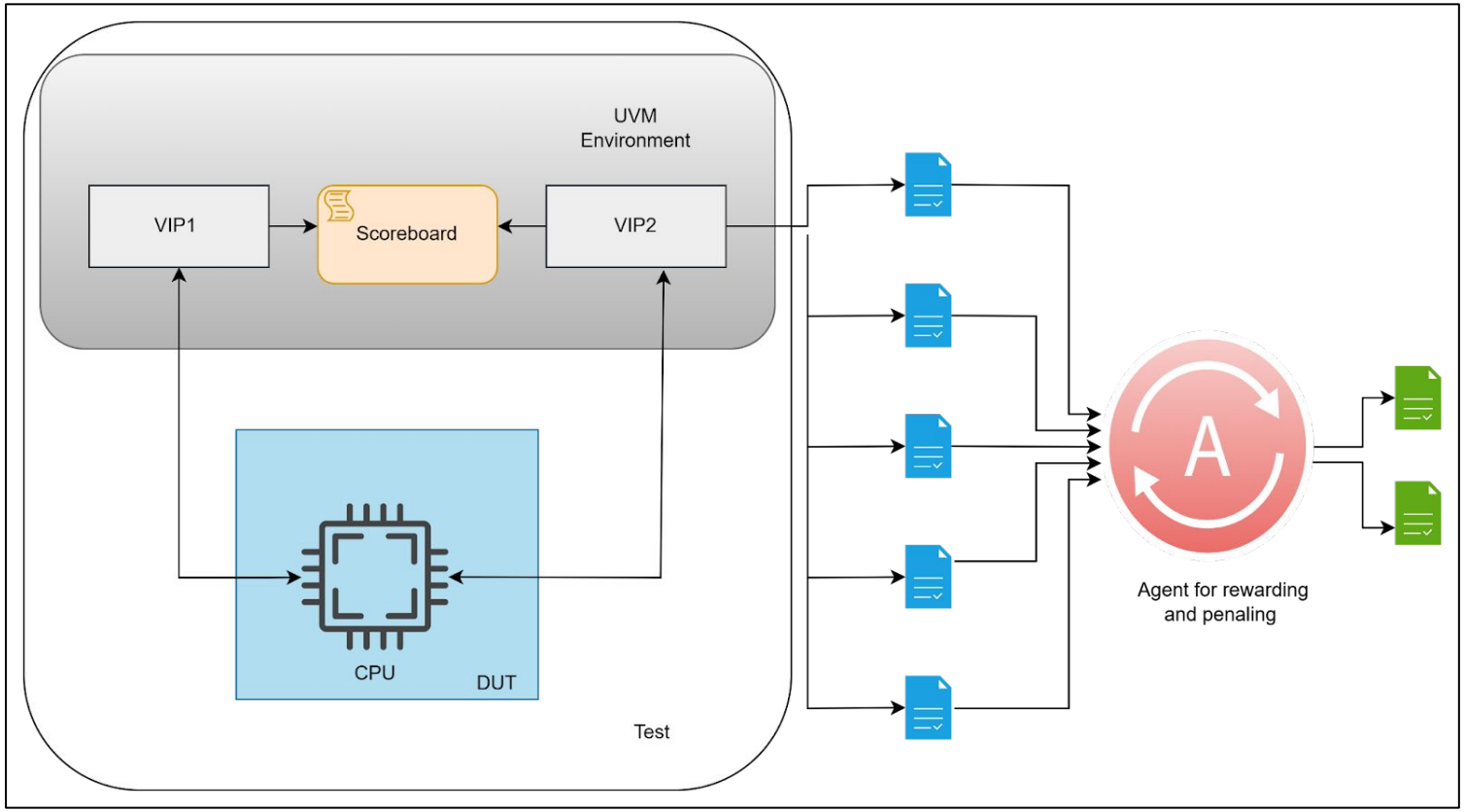
- Functional coverage closure is a bottleneck
- Random regressions inefficient
- Need intelligent guidance
- Reinforcement Learning provides adaptive optimization

Proposed Approach

- RL Agent guides stimulus selection
- Reward = new coverage
- Penalty = redundant test
- Iterative loop until closure



UVM Testbench and RL Agent Integration

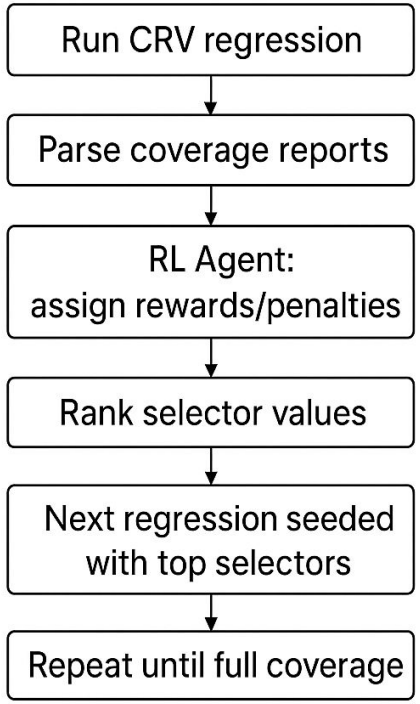


Implementation

- Agent in Python
- Minimal integration

- One random test with 3 sets of constraints
  - sel[0] - External events (Wake-up, sleep)
  - sel[1] - OverVoltage, UnderVoltage
  - sel[2] - Others

RL-Augmented Verification Flow



Pseudocode

```
for iteration in CRV_regression:
   run_test(iteration)
   covered_states = pars_coverage_report()
   if new_state_covered(covered_states):
     reward(iteration)
   else:
     penalize(iteration)

# Rank selector values based on rewards
top_selectors = rank_selectors_by_reward )

# Seed next regression with top selectors
for selector in top_selectors:
   run_test_with_selector(selector
```

Result of RL Agent processing

- The Part of Test Ranking Report

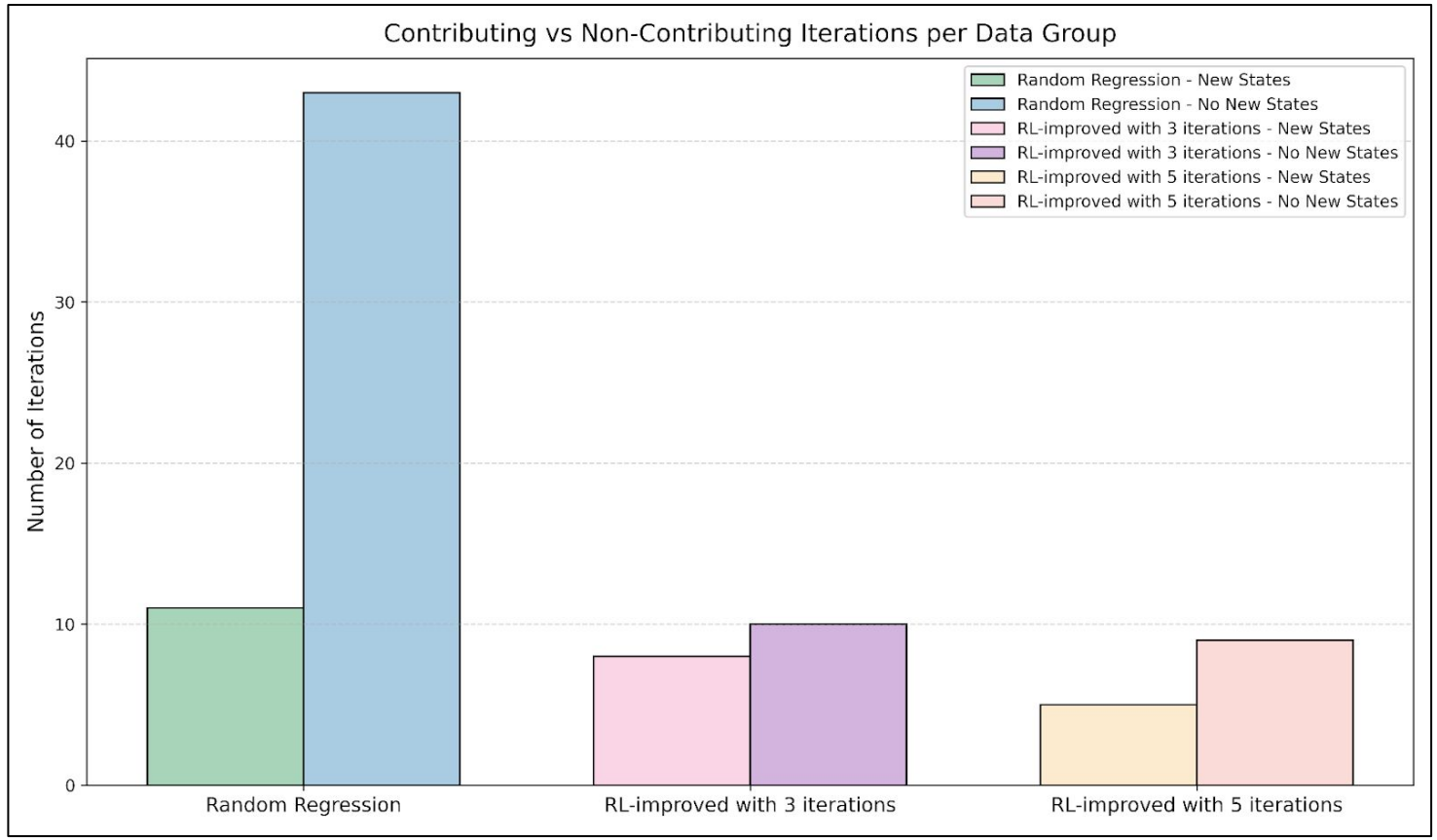
```
[NEW]         [TEST_1] State: 0000,1010               Constraints: [0, 0, 0, 0] => Reward: 1.0
[NO_NEW_COV] [TEST_2] State: /                        Constraints: [0, 1, 1, 1] => Penalty: -0.5
[NEW]         [TEST_3] State: 0100,1000,1011,1100,1111 Constraints: [0, 0, 0, 0] => Reward: 1.0
[NEW]         [TEST_4] State: 1001                    Constraints: [0, 1, 1, 1] => Reward: 1.0
[NO_NEW_COV] [TEST_5] State: /                        Constraints: [0, 1, 1, 1] => Penalty: -0.5
[NO_NEW_COV] [TEST_6] State: /                        Constraints: [0, 0, 1, 0] => Penalty: -0.5
[NEW]         [TEST_7] State: 0001                    Constraints: [0, 1, 0, 1] => Reward: 1.0
[NEW]         [TEST_8] State: 1101                    Constraints: [0, 1, 0, 1] => Reward: 1.0
[NO_NEW_COV] [TEST_9] State: /                        Constraints: [0, 1, 0, 1] => Penalty: -0.5
```

- An optimized subset of constraints that maximizes coverage efficiency: `b000, `b111, `b101, `b011, `b001

Results

- FSM design experiment
- RL reduces regressions
- RL achieves ~3× faster coverage closure on the FSM benchmark
- Single random test run is called iteration = effort unit

Efficiency Gains with RL over CRV



ROI and Discussion

- Measure = iterations (number of test runs) to closure
- Faster closure means fewer regression cycles and reduced compute cost.
- Low overhead
- Reduced regression cycles translate directly to shorter verification turnaround and compute savings.
- ROI = fewer iterations + lower compute cost

Future Work

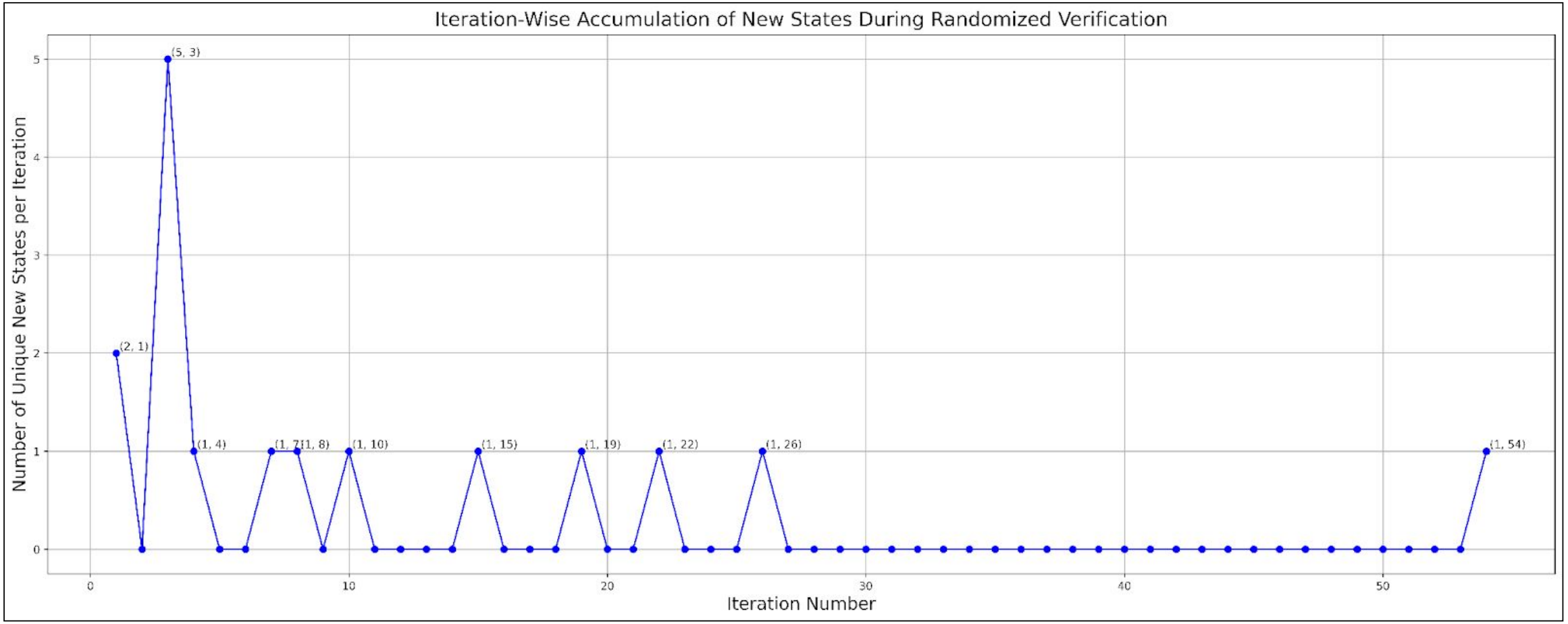
- The reward model can be extended to include:
  - Test runtime - to prefer faster tests
  - Critical features - to prioritize tests which are covering critical features
  - Diversity- to encourage test variety
- Evaluation of the proposed flow to more complex verification environments and designs

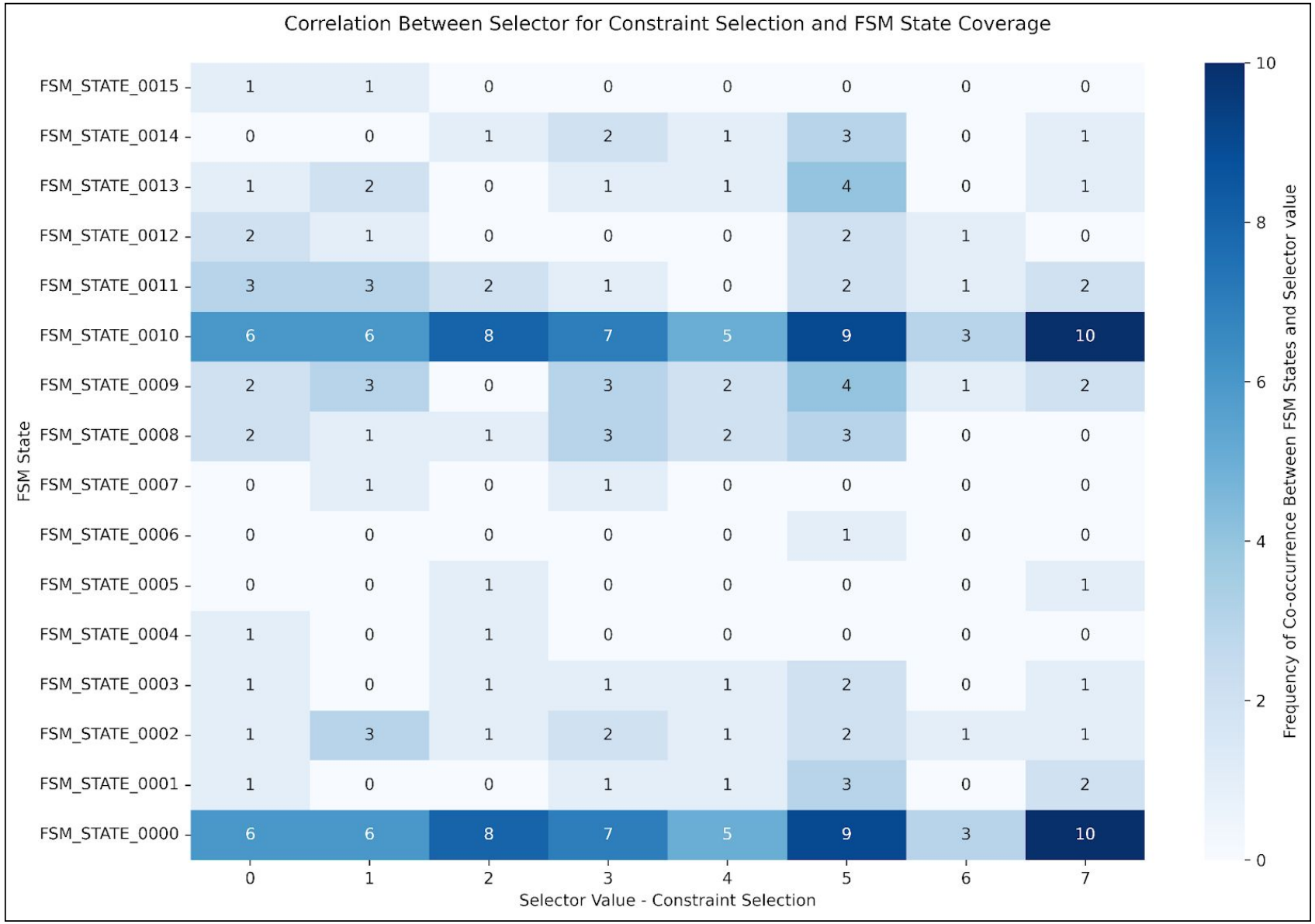
Conclusion

- RL accelerates coverage closure
- FSM case study
- Generalizable methodology
- Future: larger designs
- Key takeaway: fewer iterations = faster closure

Questions

- Thank you for your attention
- All questions and suggestions are welcome


Iteration-Wise Accumulation of New States During Randomized Verification
Number of Unique New States per Iteration
Iteration Number
(5, 3)
(2, 1)
(1, 4)
(1, 7)
(1, 8)
(1, 10)
(1, 15)
(1, 19)
(1, 22)
(1, 26)
(1, 54)
0
1
2
3
4
5
0
10
20
30
40
50


Correlation Between Selector for Constraint Selection and FSM State Coverage
FSM_STATE_0015 1 1 0 0 0 0 0 0
FSM_STATE_0014 0 0 1 2 1 3 0 1
FSM_STATE_0013 1 2 0 1 1 4 0 1
FSM_STATE_0012 2 1 0 0 0 2 1 0
FSM_STATE_0011 3 3 2 1 0 2 1 2
FSM_STATE_0010 6 6 8 7 5 9 3 10
FSM_STATE_0009 2 3 0 3 2 4 1 2
FSM_STATE_0008 2 1 1 3 2 3 0 0
FSM_STATE_0007 0 1 0 1 0 0 0 0
FSM_STATE_0006 0 0 0 0 0 1 0 0
FSM_STATE_0005 0 0 1 0 0 0 0 1
FSM_STATE_0004 1 0 1 0 0 0 0 0
FSM_STATE_0003 1 0 1 1 1 2 0 1
FSM_STATE_0002 1 3 1 2 1 2 1 1
FSM_STATE_0001 1 0 0 1 1 3 0 2
FSM_STATE_0000 6 6 8 7 5 9 3 10
FSM State
0 1 2 3 4 5 6 7
Selector Value - Constraint Selection
Frequency of Co-occurrence Between FSM States and Selector value
10 8 6 4 2 0