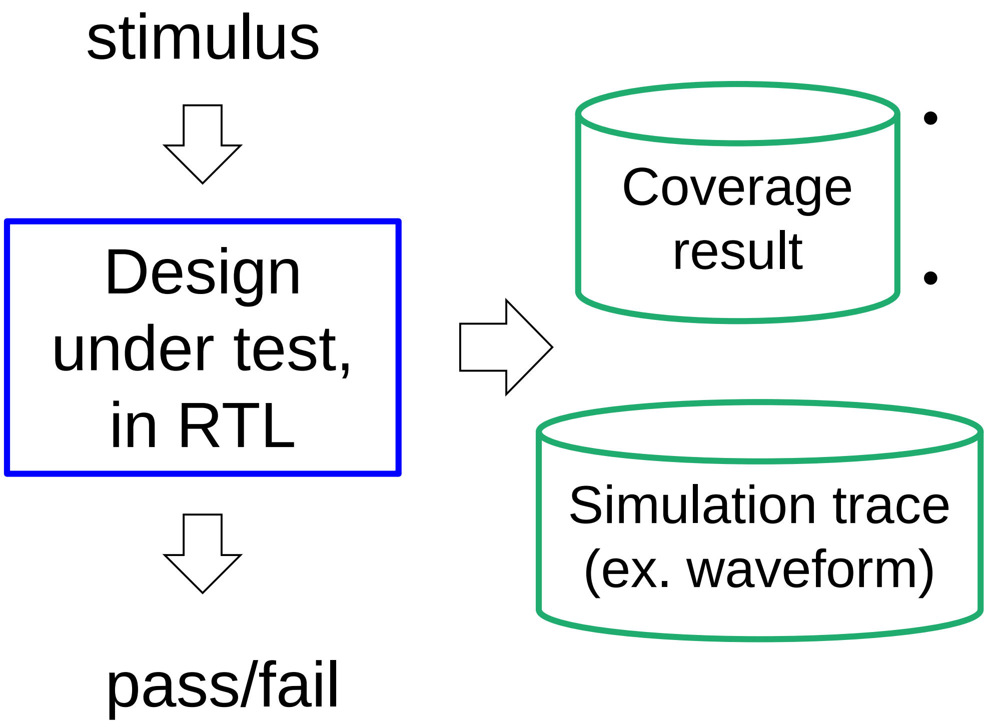
Feature Extraction from Design Documents to Enable Rule Learning for Improving Assertion Coverage

Kuo-Kai Hsieh, Sebastian Siatkowski, Li-C. Wang
University of California, Santa Barbara

Wen Chen, Jayanta Bhadra
NXP Semiconductors, Inc.

Background

- Simulation-based RTL verification



- Monitor whether certain events happened or not
- Ideally we want 100% coverage

- Record all the design behavior in simulation
- Useful for analyzing design and debug

Rule Learning Applications

- To accelerate the verification process

- Given a event, find a rule composed of design signals that can infer the event

$$!\,(Sig_A = 1) \wedge (Sig_B = 0\ to\ 1)\ \rightarrow event\ happen$$

- Application examples

Event	Application
Coverage points	Accelerate coverage closure
Bug behaviors	Triage, debug assistance

Key Challenge

- Feature/signal selection
  - The rule learning method works only for a small amount of signals
  - Limited by the amount of data
- Conceptual illustration of the issue
  - Suppose only A, B are relevant to the event E
  - Case 1, features are only A and B

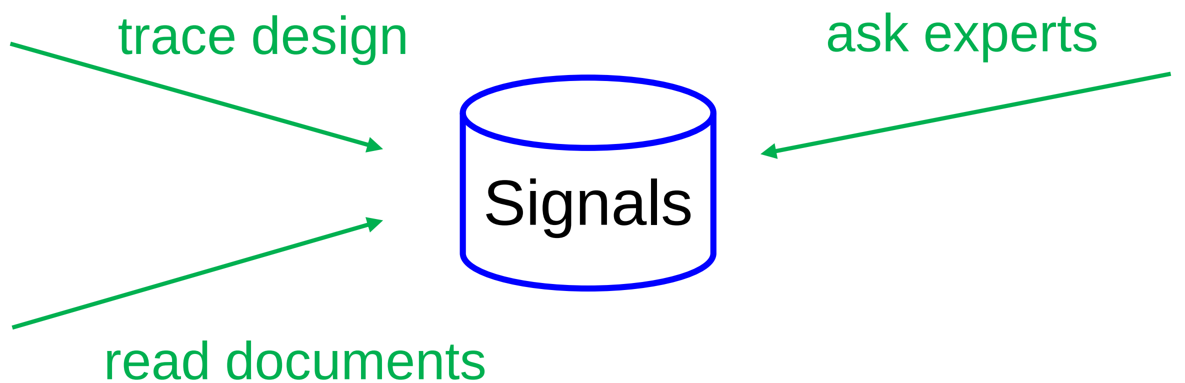
    Need 4 samples to learn E = f(A, B)
  - Case 2, an irrelevant feature C is included

    Need 8 samples to learn E = f(A, B, C)

The amount of required data grows exponentially

More Challenges

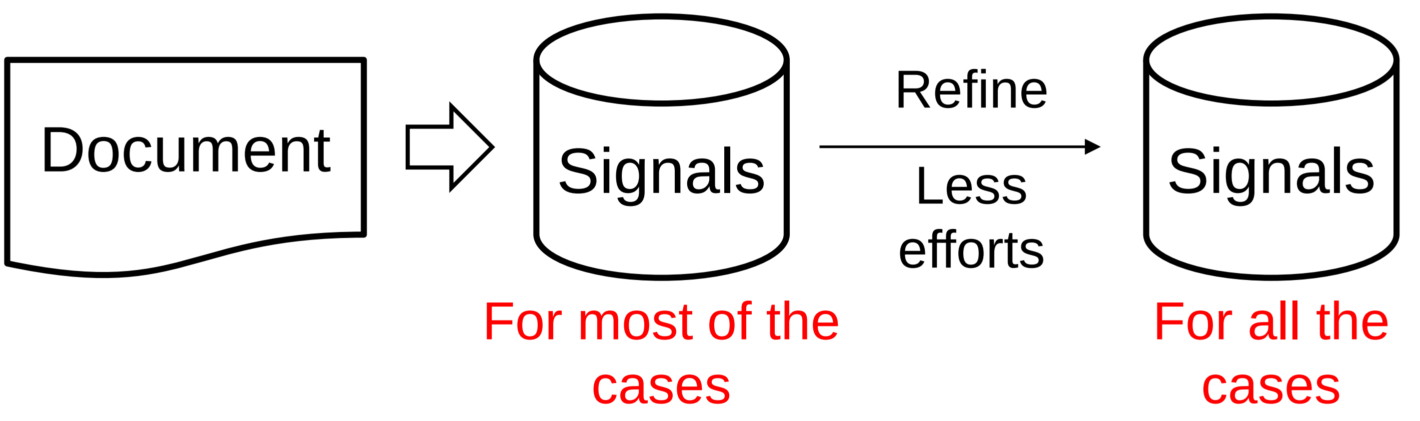
- Typically we want the selected signals to be
  - Relevant to target events
  - Understandable by human, have high-level meaning
- How people select the signals in practice?
  - It is a costly process



Extracting Signals from Documents?

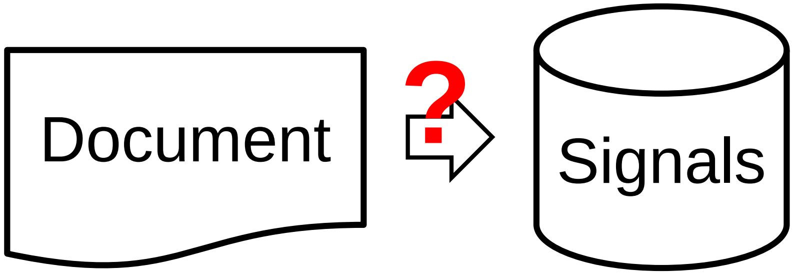
- Typically we want the selected signals to be
  - ? Relevant to target events
  - V Understandable by human, have high-level meaning

- To accelerate the process of getting the relevant signals

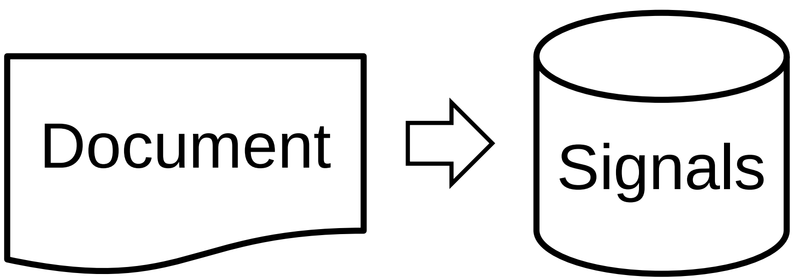


Our Problems

- How to extract signals from documents?



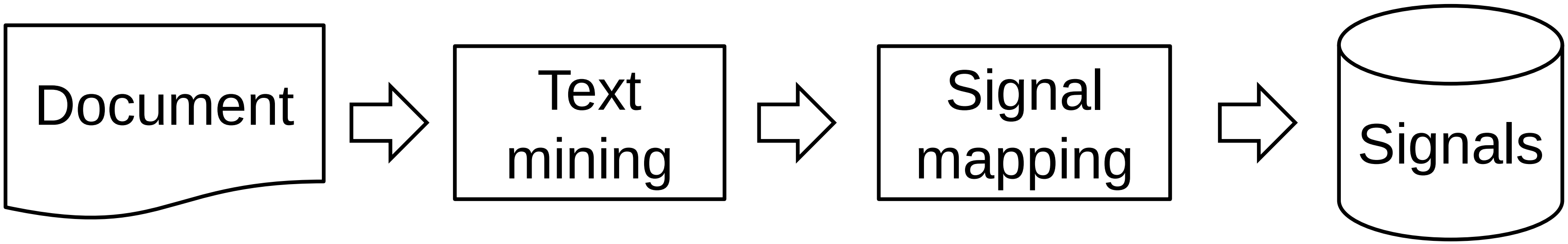
- How does the extracted signals performs?



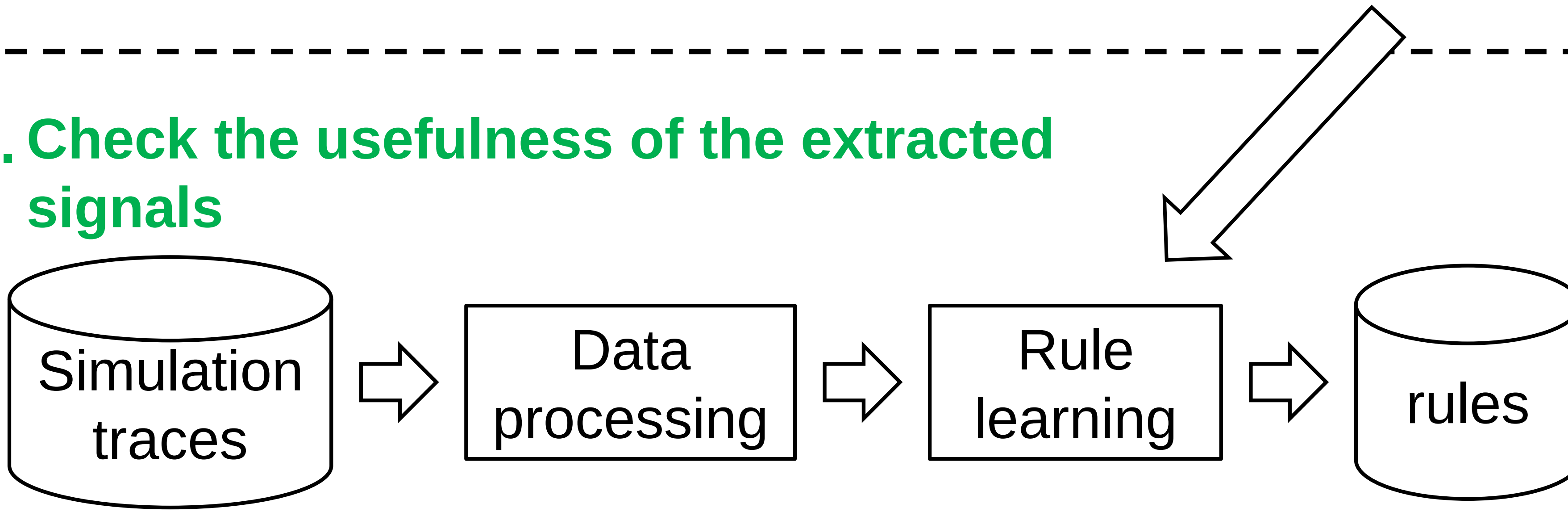
For most of the cases ?

Overview of The Proposed Method

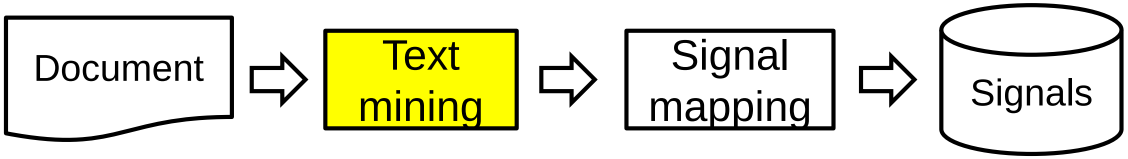
1. Signal extraction from design documents



2. Check the usefulness of the extracted signals

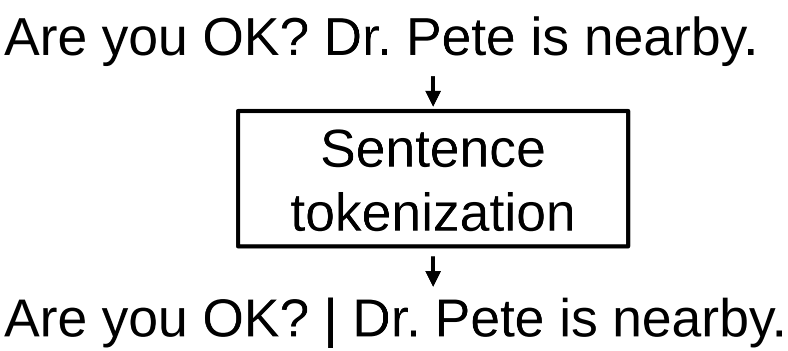


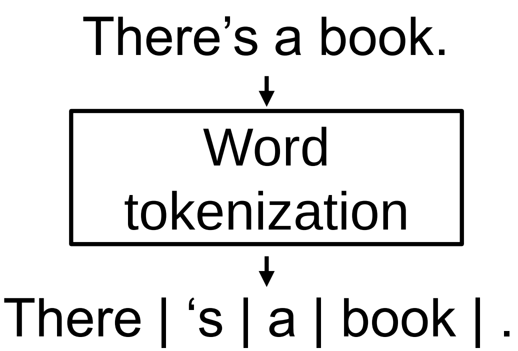
Text Mining



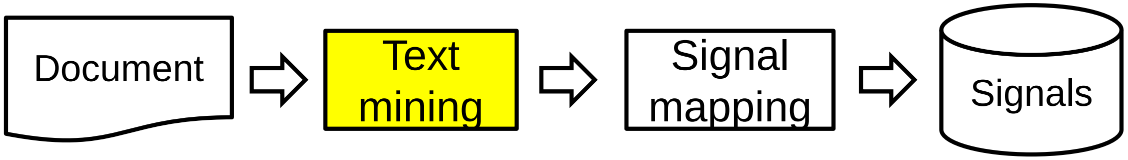
- Goal :
  - Process design documents
  - Output the words that represent design signals

1. Tokenization

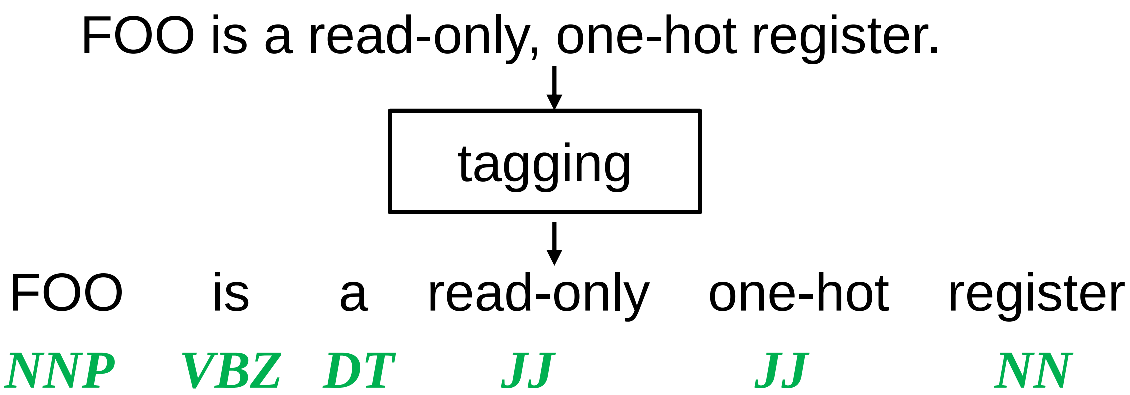




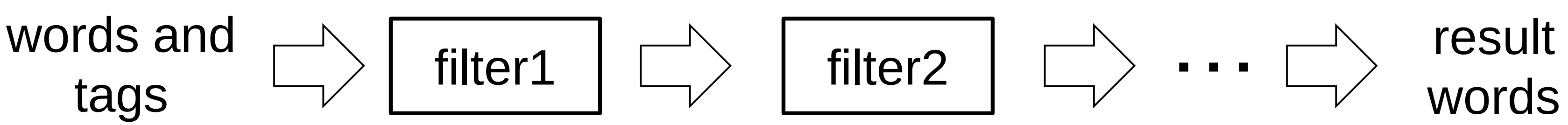
Text Mining



2. Part-of-speech tagging

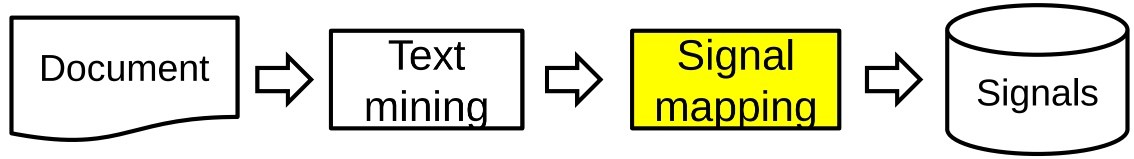


3. Application specific filtering



- Select nouns, select all-capital words, select words with underscores, remove specific words, …

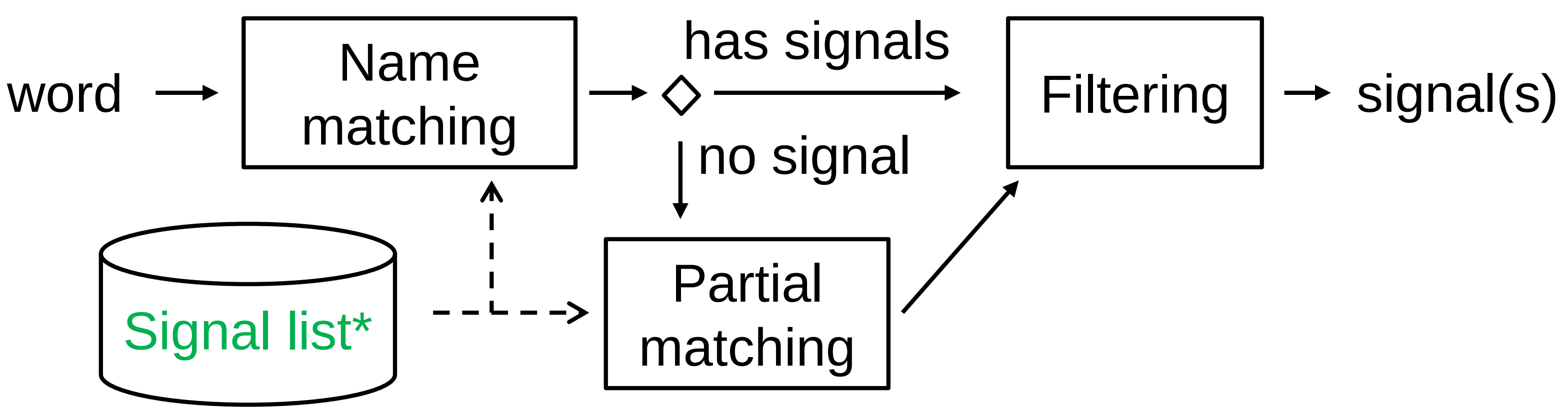
Signal Mapping



- Goal
  - Map the extracted words to the design signals, with signals' full hierarchy

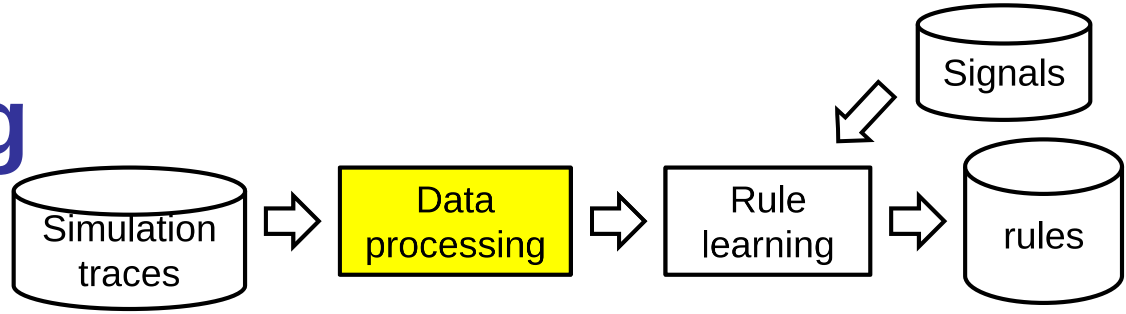
    ex. *sleep_req* → *chip_top.foo.bar.sleep_req*

- Based on string matching



* Signal list can be obtained via commercial tools

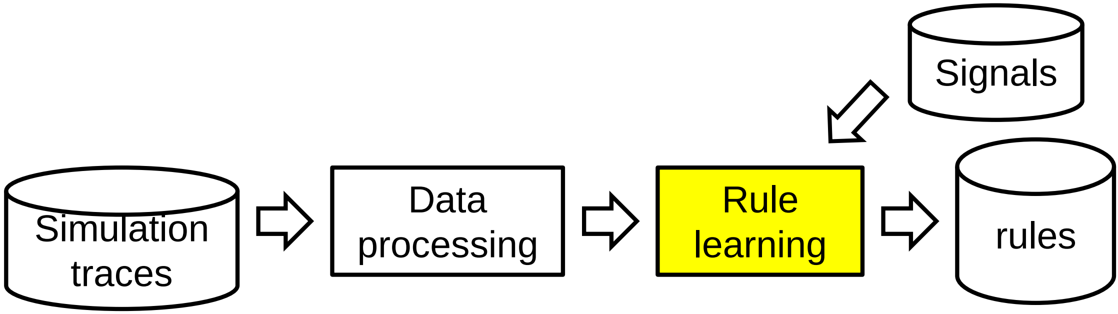
Data Processing



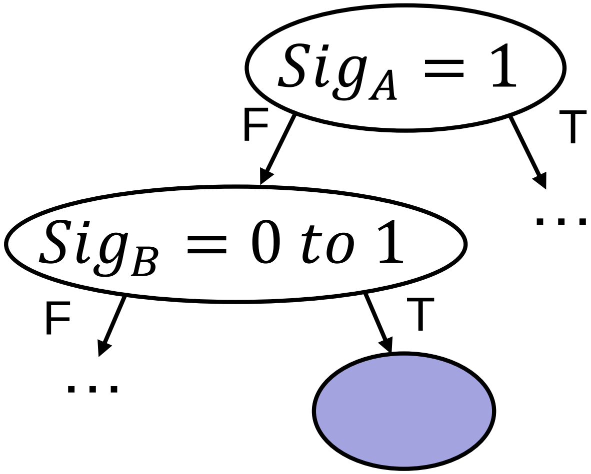
- Convert simulation waveform to a data frame ready to the rule learning algorithm
  - Remove duplicate timestamps
  - Parameters are Boolean. They include “signal = v”, “signal = v_1 to v_2”
  - A coverage event may be asynchronous to the signals.

param1	param2	…	paramN	Target cov.
1	0		1	0
0	0		1	1
0	1		1	1
1	0		0	0
1	1		0	0

Rule Learning



- Decision tree classifier
  - Iteratively find a parameter that can best split the current set of samples
  - The result rule is
    - The disjunction of all the leaf nodes having only positive sample if such nodes exist; otherwise,
    - The leaf node have the highest ratio of positive samples.



The leaf node maps to the rule:

$$!\,(Sig_A = 1) \land (Sig_B = 0\ to\ 1)$$

Experimental Environment

- Verification environment of a commercial dual-core SoC targeting ultra-low power applications
- Observed on 168 assertion coverage points
  - focusing on low-power features
  - developed by the verification engineers
- Worked on a 49-page PDF design document
  - describing low-power functionality
  - Written in natural language. No specific format.
- Data collection
  - 500 tests for training
  - Another 500 tests for validation

Signal Extraction Results

# words after text mining	71
# words having signal mapping	42
# signals of the mapping result	46

- 49-page PDF design document
- The words don't have signal mapping includes
  - The name of hardware modules
  - Design-specific abbreviation
- Observations
  - Two different words may map to the same design signal
  - A word may map to multiple design signal

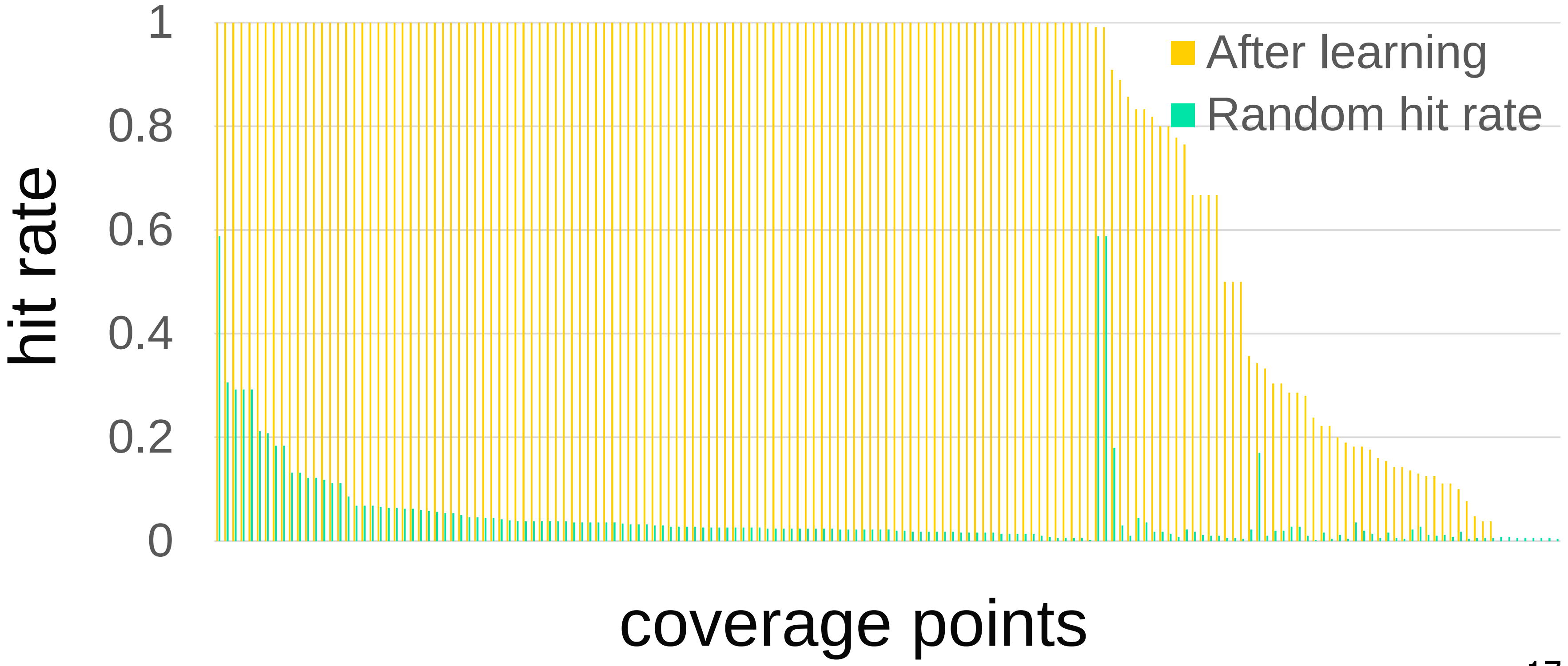
Data Processing Results

- 46 design signals
- 300 parameters
- 500 training tests contribute to 9216 training samples

- Note: there are around 240k signals in the SoC
  - Dumping all the signals and processing all of them is infeasible

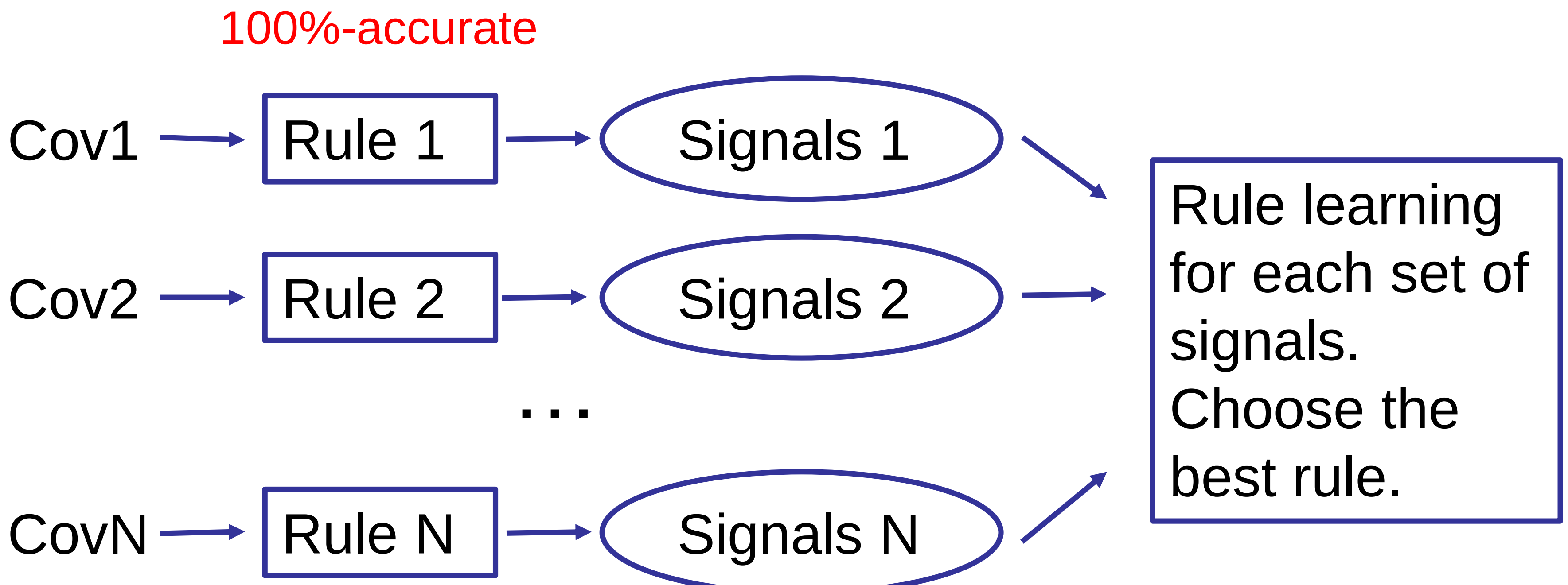
Rule Learning Results

- 168 assertion coverage points
- Obtained 100%-accurate rules for 64% assertion points



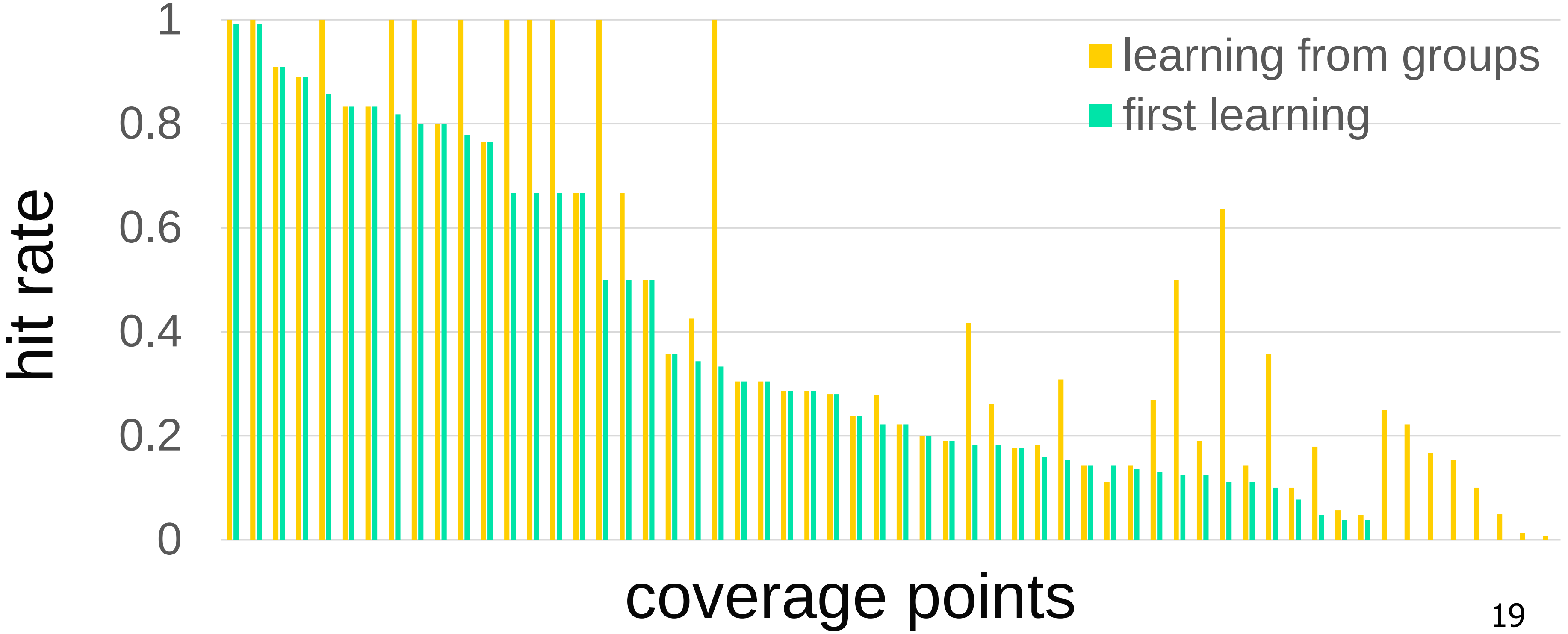
Further Improvement

- Idea – increase rule accuracy
  - learning from a smaller group of signals
  - If a set of signals can infer a coverage point, it may be able to infer another coverage point



Improved Rule Learning Results

- 38/58 coverage points have hit rate improvement
- Achieve 100%-accurate rules for 11 more points
  - Overall, 71% points have 100%-accurate rules



Conclusion

- It is expensive to apply rule learning methods without deep design knowledge.
- We proposed a signal extraction flow from the design documents.
- Experiments showed that the extracted signals can infer more than 70% coverage points with 100%-accuracy.
- The set of the extracted signals provide a good starting point. It still takes effort to deal with the rest coverage points.