RNYF

DESIGN AUTOMATION CONFERENCE 56
FROM CHIPS TO SYSTEMS – LEARN TODAY, CREATE TOMORROW

Specification Mining For SoC Validation Using Data Mining Techniques

Md Rubel Ahmed, Yuting Cao, Hao Zheng; Dept. of CSE, University of South Florida

Abstract

The quality of an SoC validation depends on the quality of specifications against which it has been tested. So effective SoC validation requires well-documented specifications. However, these specifications are often incomplete, contain inconsistencies, or even may not exist. In this work, we try to infer validation specifications from the message flow of SoC execution traces using traditional and custom data mining techniques. Message flows govern how IP blocks in an SoC design communicate with each other to realize system-level functionality. Sequential pattern mining is used along with domain specific optimization mechanisms to make the mining process more efficient and accurate. We also consider the soundness of our approach throughout the work.

Problem Statement

Our proposed approach for specification mining is done at two levels: *On-chip fabric* and *Application*.

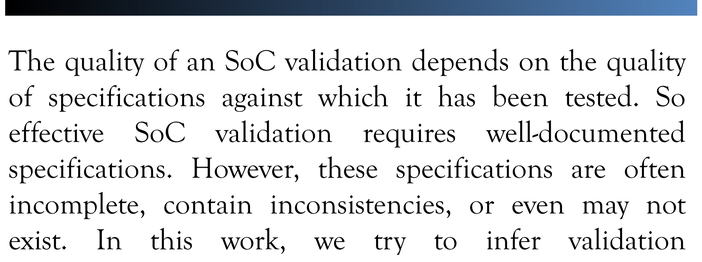

Fig. 1: Specification Mining Framework

Mine flow specification such as CPU downstream write/read etc. The fabric level specification must be valid across different execution traces as they are supposed to be implemented by the on-chip fabric. Here, we define patterns as *Sequences of events.*

The base idea came from the hypothesis that *General execution patterns can be mined from example traces of execution, which can provide correct specification for silicon validation.*

An SoC is a combination of reactive components that works together to complete a set of tasks required by the user. We characterize the patterns for mining as:

- Set of events
- Strong ordering rules among them
- In constant environment, every execution trace hold these rule

Method

Input: Execution trace encoded in a prescribed form, *T*
Output: Set of patterns described in the upper section, *P*
Data-preparation:

1. Prepare message-event mapping database:
   $\{1:\{src^1:dest^1:method^m\},\ldots..,\ m:\{src^m:dest^m:method^l\}\}$

2. Group events by clocks and prepare event database:
   $\{\{e_1,e_2,e_5\},\ldots.\{e_5\}\ldots..\{e_6,e_2,e_2\}\}\epsilon T$

3. Find unique symbols {S} and their frequency throughout the trace. For example;
   Unique Events = $\{e_1, e_2, \ldots\ldots, e_m\}$
   Event_freq: $\{e_1: 23, e_2: 45, \ldots\ldots, e_m:46\}$

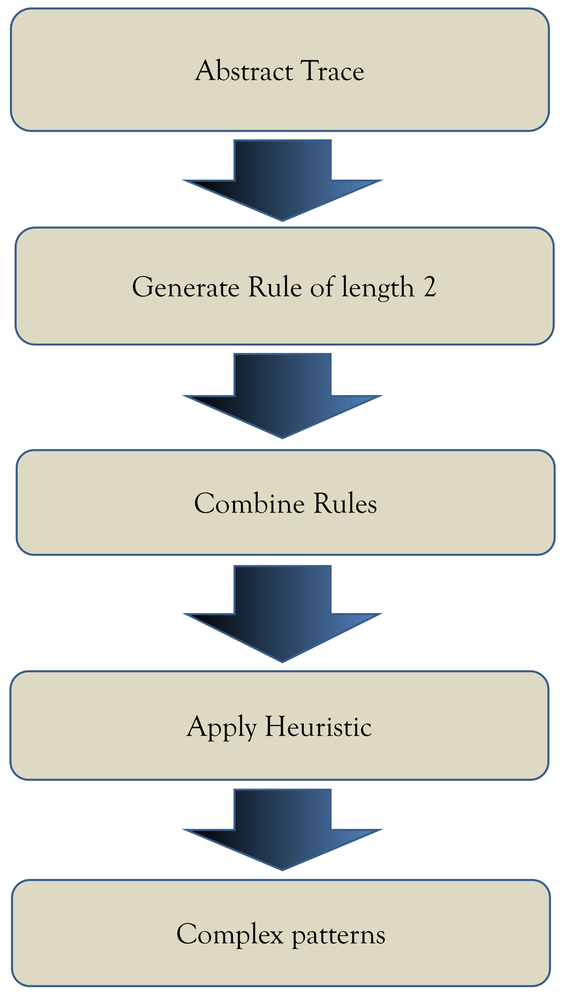

Fig. 2: Work flow of the proposed algorithm

Find rules of two events:

1. From the unique event list, take every possible pairs of two different events.
2. Generate a projected trace for each pair from the input trace that is consist of the events in the pair only. $\{e_1,e_1,e_2,e_2,e_1,e_2\ldots\ldots\ldots,e_1\}$
3. Find support for this pair or rule: find number of possible pairs from the projected trace keeping them separated by clocks. Like; $\{(e_1, e_2) :15, (e_1, e_3): 24 ,\ldots\ldots, (e_{n-1}, e_n):7\}$
4. Find confidence/recall for each pair using standard confidence calculation method.
   $\{e_{n-1}, e_n$: Confidence, Recall)$\}$

Grow rule of more events using domain heuristic:

For a rule (e_1, e_2):

1. If it has confidence of 100%, find another rule (e_2, e_3) with confidence of 100%, and create a new rule (e_1, e_2, e_3).
2. If it has recall of 100%, find another rule of (e_0, e_1) with recall of 100%, and create new rule (e_0, e_1, e_2).
3. To reduce search space further, apply following pruning strategy for rule (e_1, e_2); e_1:$\{src^1:dest^1\}$ and e_2:$\{src^2:dest^2\}$ check if $dest^1 = src^1$. If this condition does not hold, discard these rules.

Complex patterns:

1. A complex patterns should to capture a complete execution flow
2. Special measures will be adopted to get discard false patterns
3. A balance between soundness and accuracy is a must

Challenges

- False patterns
- Difficulties in branches
- Execution time
- Missing patterns
- Poor event correlation

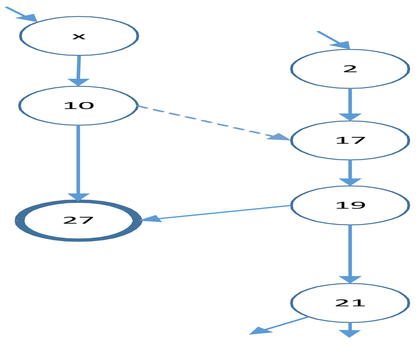

Fig. 3: Branching in flows

Proposed Solution

- Mining from sliced trace: Improves event correlation, reduces false pattern, finds missing branches
- Windowing techniques: Reduces false pattern

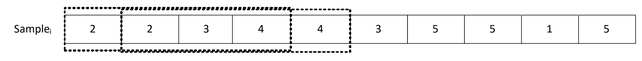

Fig. 4: Trace scanning using fixed size window

- Probabilistic branch growth: Reduces false pattern

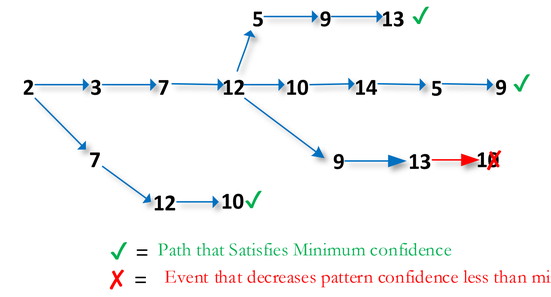

Fig. 4: Discarding branch based on minimum confidence

- Dynamic rule confidence adjustment: Reduces loosely correlated patterns.

Rule Evaluation

Let *P* be the set of all valid patterns that are known, M be the set of patterns mined using our method.

Soundness: $P \subseteq M$ holds.

Accuracy: $\frac{|P|}{|M|}$

We define Mp as the subset of patterns from M such that:

$\{ Mp \mid Mp \in P \text{ and } Mp \in M \}$

And the **soundness** of M can be defined as:

$$Soundness\ (M\_P) = \frac{|Mp|}{|P|}$$

Result Analysis

Mining sequential rule of larger length has always been a challenging task, especially for concurrent systems. One of the major problems in this task is exponential rule explosion. The no. of rules we extract for different length shows the effectiveness of our approach.

Rule Length	Permutation Method	Proposed Method
2	1806	182
3	74046	115
4	2961840	290
5	115511760	495
6	4.38944688E+9	969
7	1.624095345E+11	1538
8	5.846743244E+12	3341

Table 1: Search space comparison between proposed approach and permutation based approach

Please visit https://bit.ly/2EKpKvo for more information.