Specification Mining From Message Flow For SoC Validation

Md Rubel Ahmed, Yuting Cao, Hao Zheng
Department of Computer Science and Engineering, University of South Florida

USF 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, this 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 governs 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 through out the work.

Problem Statement

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

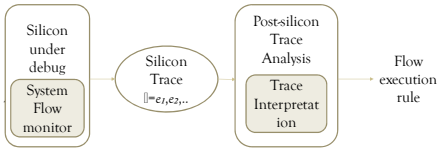


Fig. 1: Specification Mining Framework

Fabric Level

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*.

Application Level

Mine patterns among flow specifications that hold across different applications or tests. For example, the firmware loading flow should always happen after the firmware authentication flow. Hence, we define patterns as *Sequences of flows*

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 post 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

Proposed algorithm to accomplish our goal:
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:{src¹:dest¹:method□},..... ....., m:{src□:dest□:method□}}
2. Group events by clocks and prepare event database: {{e1,e2,e3},....{es}.....{e6,e2,e2}}∈*T*
3. Find unique events {E} and their frequency throughout the trace. For example; Unique events = {e1,e2, ..., em} Event_freq: {e1: 23, e2: 45, ..., em:46}

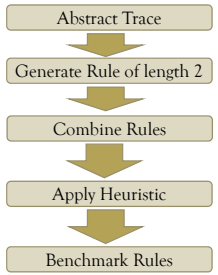
Find rules of two events:

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. {e1,e1,e2,e2,e1,e2.....,e1}
3. Find support for this pair or rule: find number of possible pairs from the projected trace keeping them separated by clocks. Like; {e1, e2 :15, e1,e3: 24,....., e(n-□),en:7 }
4. Find confidence/recall for each pair using standard confidence calculation method. {e(n-□), en : Confidence, Recall)}

Grow rule of more events:
For a rule (e1,e2):

1. if it has confidence of 100%, find another rule (e2,e3) with confidence of 100%, and create a new rule (e1,e2,e3).
2. if it has recall of 100%, find another rule of (e0,e1) with recall of 100%, and create new rule (e0,e1,e2).
3. To reduce search space further, apply following pruning strategy for rule (e1,e2); e1:{src¹:dest¹} and e2:{src□:dest□} check if dest¹ = src□. If this condition does not hold, discard these rules.

Work Flow:



Rule Evaluation:

The soundness of the algorithm is our first priority. We define soundness and accuracy of our approach using following metrics.
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:

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

The *soundness* of M over *P* defines the percentage of correct patterns mined among the total of correct patterns (*P*). As mining result's *soundness* increases, the desired patterns included in the result increases, regardless of the total number of patterns (noise patterns) generated.
On the base of the *soundness*, for pattern sets M with high *soundness*, we use accuracy to evaluate its ability of filtering out irrelevant, unimportant patterns.

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

The *accuracy* M over *P* defines the percentage of correct patterns mined among all patterns in M. A mining algorithm that produces high accuracy result means it can generate as set of correct patterns with minimal noises.

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. When we have a large number of events to consider and if we want to find rules of higher length given that the order is preserved, we face the problem of being out of enough drive space or some hrs of running time. For these reason, with the best of our knowledge, no specification mining work have been done so far that mines exact ordered rules for concurrent hardware systems. In the works of episode mining it takes account of a series of precedent events and mines a series of consequent event for them. The fundamental difference between episode mining and our approach is that we can not compromise a single event order. We mine strict ordering relations among the events. We try to find rule violations for SoC internal communication protocols. In practical it is common that some abnormal behavior exposes only single time in tons of events. So one such event can drag the whole system into failure. For these issues we could not apply episode mining for our problem. Another common strategy in finding sequential rule is "sliding window"

Rule Length	Permutation Method	Proposed Method
2	1806	1806
3	74046	7360
4	2961840	14720
5	115511760	29440

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

technique. This technique is very popular and efficient for search space reduction. But we can not utilize this technique in our algorithm for the reason that, some flow may have a huge time distance between start event and the termination event. Sliding window can find never such episode because of this distance though it is a valid rule for many of the hardware systems. So we employ different event selection technique based on confidence, recall and heuristic to ensure that no valid rule is skipped in our mined pattern. That's how we ensure the soundness of our algorithm.

References

[1] Sandip Ray, Ian G. Harris, Goerschwin Fey, and Mathias Soeken. Multilevel design understanding: From specification to logic invited paper. In *Proceedings of the 35th International Conference on Computer-Aided Design*, ICCAD '16, pages 133:1–133:6, 2016.

[2] W Chen, S. Ray, J Bhadra, M Abadir, and Li-C Wang. *Challenges and trends in modern soc design verification*. IEEE Design Test, 34(5):7–22, Oct 2017.

[3] Matthew B. Dwyer, George S. Avrunin, and James C. Corbett. Patterns in property specifications for finite-state verification. In *Proceedings of the 21st International Conference on Software Engineering*, ICSE'99, pages 411–420, 1999.

[4] Glenn Ammons, Rastislav Bod´ık, and James R. Larus. Mining specifications. In Proceedings of the 29th ACM *SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '02, pages 4–16,2002.

[5] Po-Hsien Chang and Li.-C Wang. Automatic assertion extraction via sequential data mining of simulation traces. In *Proceedings of the 2010 Asia and South Pacific Design Automation Conference*, ASPDAC'10,pages 607–612, 2010.

[6] Wenchao Li, Alessandro Forin, and Sanjit A. Seshia. Scalable specification mining for verification anddiagnosis. In *Proceedings of the 47th Design Automation Conference*, DAC'10, pages755–760, New York, NY, USA, 2010. ACM.

[7] Samuel Hertz, David Sheridan, and Shobha Vasudevan. Mining hardware assertions with guidance fromstatic analysis. *Trans. Comp.-Aided Des. Integ. Cir. Sys.*, 32(6):952–965, June 2013.

[8] A. Danese, F. Filini, and G. Pravadelli. A time-window based approach for dynamic assertions mining oncontrol signals. In *2015 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*,pages 246–251, Oct 2015.