Message Flow Mining for SoC Validation for Safe and Secure IoT Edge Node Design

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

Warren B. Nelms Institute for the Connected World
UNIVERSITY of FLORIDA

USF UNIVERSITY OF SOUTH FLORIDA
SEES
The SEES Lab

Introduction

SoC is the hardware foundation for IoT edge nodes. Ensuring security properties such as *confidentiality* and *integrity* is crucial for the trustworthiness of IoT devices. However, due to the high complexity of the global supply chain, ensuring trustworthiness of diverse third-party suppliers becomes very much challenging. So a comprehensive validation is needed in this regard. On the other hand, well-defined specifications are necessary to perform rigorous and thorough validation of SoC designs. However, in practice, such specifications are hardly available, often incomplete and ambiguous[1]-[7]. In this work, we aim to address such a challenge by proposing a sequential pattern mining framework, *FlowMiner* to automatically extract message flow specifications. We also propose several domain specific optimization techniques to boost up the run time of the framework. The extracted message flows characterize the communication behavior among the components of a SoC design, thus can be used in validation and debug of IoT edge nodes. We evaluate our framework on execution traces generated from simulation of a non-trivial multi-core SoC design model and on a set of complex synthetic traces. We evaluate extracted sequential patterns in terms of precision and recall. Our framework shows better result in these metrics compared to another benchmark temporal property miner called *Perracotta[8]*.

Background

An SoC is a combination of reactive components, called IPs that work together to complete a set of tasks. The IPs follow some system level protocols. Many experiments have shown that the implementation of those system-level protocols is the major sources of various design errors. Therefore, communication-centric validation is a key activity of SoC validation.

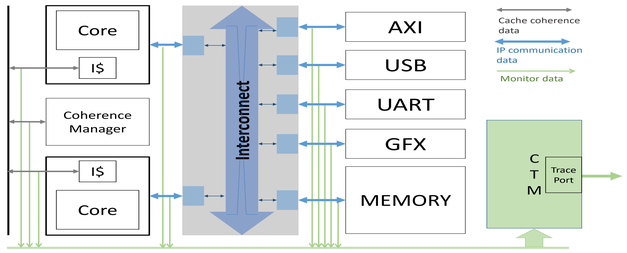

Fig. 1: An SoC prototype with different IPs

We can view a task as a message flow specification, for example, CPU downstream write.

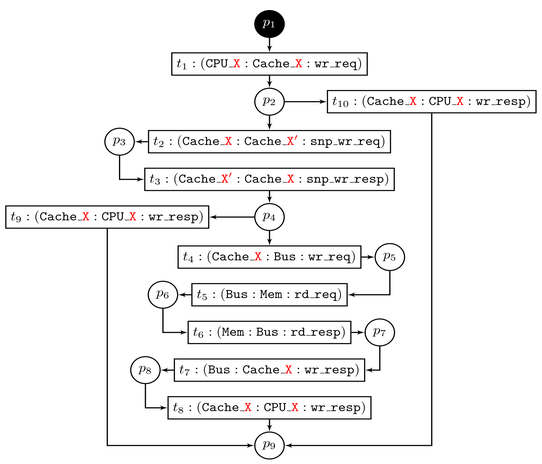

Fig. 2: LPN formulation of a CPU write flow

Problems Addressed

Silicone validation is becoming more and more challenging with the advent of more complex and customized hardware. This framework aims to address:

1. Post-silicon validation
2. Specification mining
3. False positive specification
4. Specification mining time

We characterize the patterns to be mined as:

- ❑ Set of events
- ❑ Strong temporal dependency
- ❑ In constant environment, each execution holds the rules

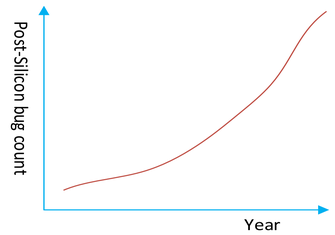

Fig. 3: Cost of Silicon validation getting worse (source: intel)

Proposed Algorithm

We utilize association rule mining technique to mine sequential patterns from the execution traces. We also apply domain specific heuristic to reduce the huge search space of association rules. Proposed algorithm works on execution traces captured by monitoring the messages among the IPs of an SoC.

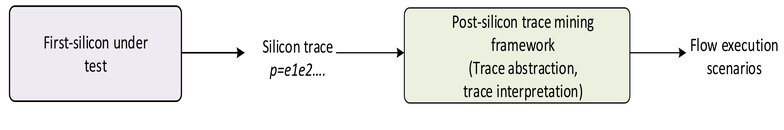

Fig. 4: Post-silicon trace mining

FlowMiner workflow

Trace Processing: We perform **address-based trace slicing** in this step. Trace processing is an important step where the principle focus is to increase correlation between the base patterns which are used to form longer patterns in the later steps. We utilize address data field of each message found in the trace to slice original trace into sub-traces and mine binary patterns from them in the next step.

Pattern mining: We apply 100% forward and backward confidence restriction to mine patterns that hold over all traces. The reason to mining assertion is to find out flows implemented by the IoT hardware core, which are invariant over different trace. This strong strictness in this step also reduces the overall search space without compromising our sole objective. The outcome of this step two set of patterns where each of the subsequences has 100% *forward* or *backward confidence* or *both* at the same time.

Forward Confidence: For a sequence $e_i \rightarrow e_j$ in a trace T, $conf_f((e_i, e_j), T) = \frac{support((e_i, e_j), T)}{support(e_i, T)}$

Backward Confidence: For a sequence $e_i \rightarrow e_j$ in a trace T, $conf_b((e_i, e_j), T) = \frac{support((e_i, e_j), T)}{support(e_j, T)}$

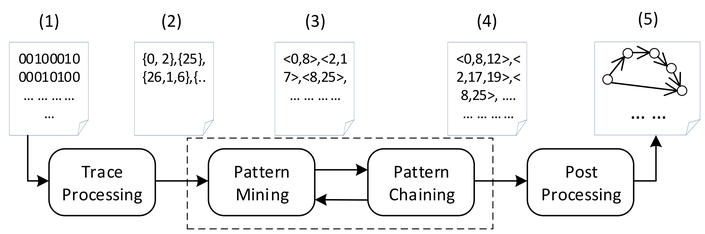

Fig. 5: Worflow of *FlowMiner*

Pattern chaining: One of the fundamental problems of rule-based systems is the huge search space complexity. After mining for each iteration, we apply 3 inference rules to generate longer patterns to avoid costly mining process. This is one of our optimization steps.
The iteration between mining and chaining keeps going until all the valid rules are found upto a user defined length *L*.

Post Processing: This is the representational and optional step. In this step mined patterns could be ranked according to the user demand. It also counts the number of branches, most-often taken branches etc. in the flows.

Results

	# Patterns Mined	Precision	Recall
FlowMiner	14	1.0	1.0
Perracotta	12	1.0	0.0

Tab. 1: Flow mining from synthetic traces of **CPU0_write**

	# Patterns Mined	Precision	Recall
FlowMiner	316	0.66	0.19
Perracotta	127	0.47	0.06

Tab. 2: Flow mining from synthetic traces of 10 different flows

	# Patterns Mined	Precision	Recall
FlowMiner	21	0.95	0.57
Perracotta	8	1.0	0.12

Tab. 3: Flow mining from SoC model simulation execution traces

Mining sequential patterns of longer lengths has always been a challenging task, especially for concurrent systems that are also recurrent. Hence our proposed algorithm shows promising result compared to Perracotta. We define **precision** as the ratio between the number of valid patterns mined and the number of total mined patterns. And **recall** as the ratio between the number of original flow instances mined and original flow instances exercised.

Challenges to overcome:

- ❑ High number of false positive patterns
- ❑ Difficulties with branches
- ❑ Long execution time
- ❑ Need for perfect traces
- ❑ High concurrency yields poor correlation

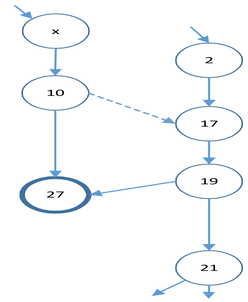

Fig. 6: Branch in flows

Conclusion

We mine strict ordering relations among the events and present them in sequential pattern form that represents the message flow specifications. The mined patterns will help to find violations for SoC internal communication protocols. Proposed framework can play an important role to make the task of IoT edge node verification easier.

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] Jinlin Yang, David Evans, Deepali Bhardwaj, Thirumalesh Bhat, and Manuvir Das. Perracotta: mining temporal api rules from imperfect traces. In Ohio University, 2006.

Please scan for more information: