Model Synthesis for Communication Traces of System-on-Chip Designs

Submitted to GLSVLSI-2021

Presenter: Md Rubel Ahmed

UNIVERSITY of SOUTH FLORIDA

Motivation

- System level model is important for SoC design analysis, and validation
- Inferring execution model from SoC communication traces is challenging because:
  - Prevalent concurrency in the traces
  - Lack of observability
  - False dependency due to parallelism

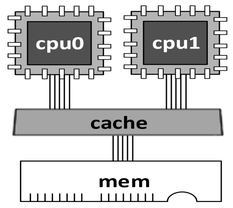


Fig. 1: A simple SoC with two cores and other peripherals

Message types

1 (cpu0:cache:rd_req)
2 (cache:cpu0:rd_resp)
3 (cpu1:cache:rd_req)
4 (cache:cpu1:rd_resp)
5 (cache:mem:rd_req)
6 (mem:cache:rd_resp)

Example flows

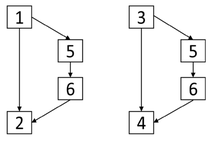


Fig. 2: CPU downstream read flows(*up*), and a sample execution trace (*below*)

({1, 3}, 1, 2, 5, 1, 5, 6, 2, 4, 6, 2)

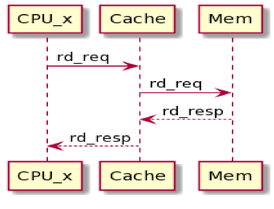
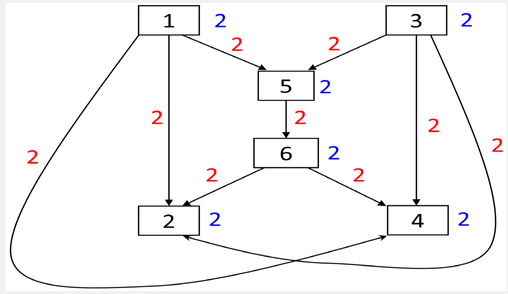


Fig. 3: Model synthesis goal

Method

1. Building Causality Graph



2. Solving Constraints

For each node n and outgoing edge $n \to n'$

$$Sup(n) = \sum_{all\ n \to n'} c(n \to n')$$

For each node n' and outgoing edge $n \to n'$

$$Sup(n) = \sum_{all\ n \to n'} c(n \to n')$$

For each edge $n \to n'$

$$0 \le c(n \to n') \le Sup(n)$$

3. Deriving Model

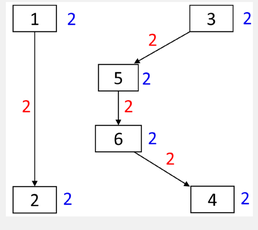


Fig. 4: Solutions suggested by Z3[2]

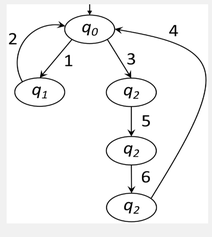


Fig. 5: Reduced model (FSA)

Experiments & Results

Traces	#Messages	Length	#States	Runtime(s)
small	22	460	31	84
		920	31	78
		1840	31	70
Large	60	2180	92	75
		4360	87	72
		8720	100	62

Table. 1: Models from synthetic traces

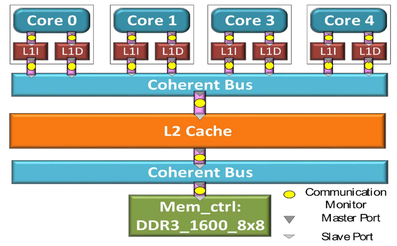


Fig. 6: Simulation trace generation on GEM5

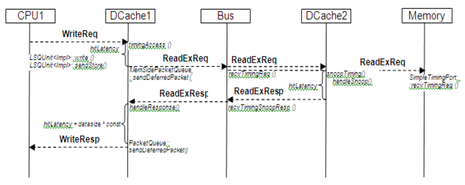


Fig. 7: Message flow specification in GEM5 documentation [1]

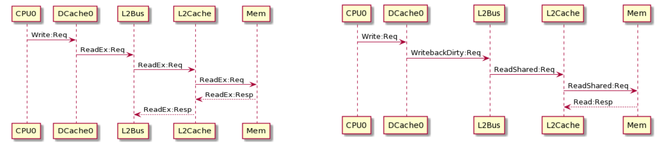


Fig. 8: Two flows from the synthesized models

Future Directions

1. Pattern reduction
2. Incorporate user's insight to guide the search

Authors:
Dr. Hao Zheng[1]
Md Rubel Ahmed[1]
Parijat Mukherjee[2]
Mahesh C. Ketkar[2]
Jin Yang[2]
1: USF, 2: Intel®

visit our lab page ☺

References:
1. Memory System in gem5.http://pages.cs.wisc.edu/ swilson/gem5-docs/gem5MemorySystem.html. Online; accessed November 16, 2020
2. The Z3 Theorem Prover. https://github.com/Z3Prover/z3, 2020. Online; accessed 17 November 2020.

Retrieve preprint ☺