DAC YF

Model Synthesis for Communication Traces of Systems Designs

Md Rubel Ahmed, Hao Zheng, University of South Florida

Motivation

- System model can help 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

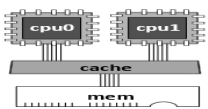


Fig. 1: A simple SoC with two cores and other peripherals

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

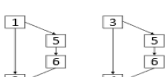


Fig. 2: CPU downstream read flows(*up*), and a sample execution trace (*below*)

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

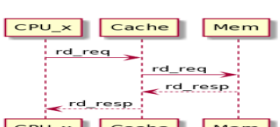
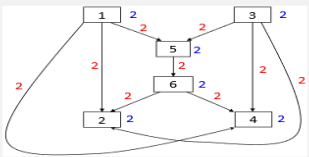


Fig. 3: Model synthesis goal

Method

1. Building Causality Graph



2. Solving Constraints

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

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

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

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

For each edge $n \rightarrow n'$

$$0 \leq c(n \rightarrow n') \leq Sup(n)$$

3. Deriving Model

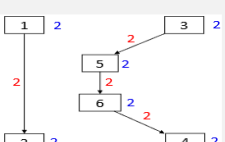


Fig. 4: Solutions suggested by Z3[2]

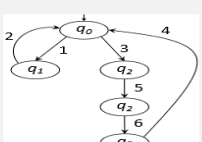


Fig. 5: Reduced model (FSA)

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	82

Table. 1: Models from synthetic traces

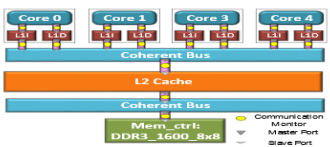


Fig. 6: Simulation trace generation on GEM5

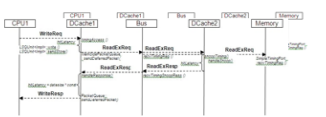


Fig. 7: Message flow specification in GEM5 documentation [1]

Fig. 8: Two flows from the synthesized models

Future Directions

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

Published on ICCD'2021