Multicore Computing – CS 5966 / 6966 – Week 5
http://www.eng.utah.edu/~cs5966

Contents

1 Overview of this week

On Monday I lectured on ARCHTEST, Test Model Checking, and Post-Silicon Verification. I also gave the class handouts on the x86 memory ordering. The list of URLs of interest are

- Intel Software Developer's Manual. Chapter 7 talks about the x86 memory model.
- The Test Model-Checking paper
- Links to ARCHTEST
  - Overview of ARCHTEST
  - Analysis pertaining to ARCHTEST
  - How to run ARCHTEST
  - ARCHTEST's Output
- Paper on Post-Si Verification may be found here.
- Slides shown in class on 2/9
- Slides shown in class on 2/11

2 Assignment on ARCHTEST

ARCHTEST administers tests on the nodes of a multiprocessor and tries to reveal its architectural rule 'weakenings.' Here, I am attempting to give only a high level overview of its operation and theory. For details, kindly peruse Collier's text-book "Reasoning about Parallel Architectures," Prentice-Hall, 1992.

2.1 How to run ARCHTEST

Running ARCHTEST is easy:

- Make a directory to run ARCHTEST (e.g. `my-archtest`)
- Make a link to all the archtest binaries (e.g., `archtest-buckley`) you see in `~cs5966/bin`.
- Copy over the parmfile into your `my-archtest` directory.
- Then be logged into machine buckley to run `archtest-buckley`, etc.
- When so logged in, simply fire-up `archtest-buckley` (or for other machines, the same idea holds)
- Then hit 'enter' for default input files etc. that ARCHTEST prompts you with.
- When the full main menu is displayed, type `3 1` by which you are selecting 'real multiprocessor test'
- Then type `4 4` by which you are selecting to run `4` threads
- Then type `2` by which you are selecting to run all the default tests that ARCHTEST is equipped to run (you may pare down this list if you wish; no harm running it all)

Then when ARCHTEST finishes, look at `a_____00.htm` using a browser. The tests being administered are described in the above HTML file as, for example,

`A(CMP,UPO,URR,WW)`

2.2 Explanation of ARCHTEST's Rules

Here, `CMP` is "computational order" and is almost never relaxed, except in a broken machine. It accounts for the normal uni-processor rules of avoiding uni-processor hazards. It also includes obeying multiprocessor cache coherence. However, UPO, as defined by Collier, also stems from obeying these. In a nutshell, CMP and UPO can be assumed to be met by modern machines.

There are also three cache-coherence rules

CC1: All threads see each operand change value at exactly the same instant. CC1 is also called write atomicity (WA).

CC2: All threads see exactly the same sequence of changes of values for all operands. CC2 is also called write synchronization.

CC3: All threads see exactly the same sequence of changes of values for each separate operand.

CC4: Different threads may see an operand assume different sequences of values.

By writing "`A(CMP,X,Y,Z)`," the author implies that a violation for `X` or `Y` or `Z` is being sought. One will have to find out what actually got violated separately (*e.g.* if `A(CMP,X,Y,Z)` is violated and so is `A(CMP,X,Y)`, then `Z` is likely not a violation. Not much else can be concluded.

We can learn ARCHTEST's encoding of events by studying **UPO** (uniprocessor ordering) in detail. Also note that in archtest, for a simple assignment such as `A=1` performed on an N-processor

system, there are N+1 events – N store events (one per store) and one read event for reading 1. Events are illustrated below (hopefully clear through examples – see Collier's book for details).

The event template for UPO is

$$\texttt{(P,L,A,V,O,S)} <_{UPO} \texttt{(=,-,-,-,=,=)}$$

This says that UPO maintains ordering between two events of the same program (P), for the same operand (O), and in the same store (S), not caring about the label (L), action - read or write - (A), or value involved (V).

Now we summarize the other event templates.

UPO: $\texttt{(P,L,A,V,O,S)} <_{UPO} \texttt{(=,-,-,-,=,=)}$

URR: $\texttt{(P,L,R,V,O,S)} <_{URR} \texttt{(=,-,R,-,=,=)}$

URW: $\texttt{(P,L,R,V,O,S)} <_{URW} \texttt{(=,-,W,-,=,=)}$

WW or WO: $\texttt{(P,L,W,V,O,S)} <_{WW} \texttt{(=,-,W,-,-,-)}$

RR: $\texttt{(P,L,R,V,O,S)} <_{RR} \texttt{(=,-,R,-,-,-)}$

WR: $\texttt{(P,L,W,V,O,S)} <_{WR} \texttt{(=,-,R,-,-,-)}$

RW: $\texttt{(P,L,R,V,O,S)} <_{RW} \texttt{(=,-,W,-,-,-)}$

PO: $\texttt{(P,L,A,V,O,S)} <_{PO} \texttt{(=,-,-,-,-,-)}$

Assignment 5 (due on 2/18): (Discussion key:) Weakenings on Machines:

Run ARCHTEST on as many machines as you can. The TA will help you obtain binaries for machines that you may have (e.g., allow him to login, he'll compile and leave the binaries behind). If any ordering relaxation is reported on a particular machine, please discuss the nature of the relaxation in a paragraph, and also the distance "d" computed by ARCHTEST. Mention which rule is being relaxed and why you think ARCHTEST's reporting seems right.

3 Assignment on the x86 Memory Model

Assignment 5 (due on 2/18): (Discussion key:) x86-memory-model-Fig-7.1

Read Sections 7.2 and 7.3 of the Intel manual I pointed out

()

Describe the orderings that you see in this figure in terms of Collier's rules, as best as you can (e.g., WW, WR, etc.)

4 Assignment on Test Model-Checking

Assignment 5 (due on 2/18): (Discussion key:) Test model checking

How is a test automaton designed for a simple (CMP,RO,WO) test? Describe the steps in a paragraph.

5 Assignment on ISP

Assignment 5 (due on 2/18): (Discussion key:) EuroPVM08-ISP

Read the paper

`http://www.cs.utah.edu/formal_verification/publications/conferences/pdf/europvm-mpi08-isp.p`

Point out, in one paragraph each, how out-of-order completion and wildcard matches are handled in ISP.

Try to Google and find out the meanings of MPI functions that you see discussed.

Assignment 5 (due on 2/18): (Discussion key:) EuroPVM08-FIB

Read the paper

`http://www.cs.utah.edu/formal_verification/publications/conferences/pdf/europvm-mpi08-barri`

Describe, in one paragraph, how MPI's barrier semantics complicates the identification of functionally irrelevant barriers.

You will understand a ton about MPI by studying these short examples. Write a paragraph describing what you observe from these examples, as well.

Assignment 5 (due on 2/18): (Discussion key:) ISP-Expt-on-Raven

Go through the entire set of instructions for the Raven machine posted at

`http://www.cs.utah.edu/formal_verification/ISP-release/isp-user-manual.pdf`

and experiment with the example (`any_src-can-deadlock9.c`) illustrated in this manual in detail, learning the use of the ISP tool in the process.

Discuss what you see to be different about ISP's algorithm (especially relating to your understanding from the EuroPVM08-ISP paper).

6 Assignment on MCAPI

Assignment 5 (due on 2/20): (Discussion key:) MCAPI-intro

Read

`http://www.multicore-association.org`

and also attend Subodh's lecture on 2/18. Based on these, summarize, in a paragraph, the salient features and intended usage of MCAPI.