A SAT-Based Procedure for Verifying Finite State Machines in ACL2

Warren A. Hunt, Jr. and Erik Reeber

{reeber,hunt}@cs.utexas.edu

The University of Texas

ACL2 Workshop, August 16·2006
Introduction

• The ACL2 theorem prover is great
  – Scalable to large industrial verification problems
• But…
  – Proofs require a lot of human effort
  – Computer could do more
    • Especially when in decidable domains
• Identify a decidable subclass of ALC2 properties
  – Based on tree structures
• New ACL2 hint for proving properties in this domain
  – Available in a future version of ACL2
Presentation Overview

• Focus is on what you can do with the new hint
  – If you want to know how it works
    • Read paper
    • Look at code: www.cs.utexas.edu/users/reeber

• Outline
  – Demo
  – Performance Results
  – Hardware Verification Methodology
  – Application: TRIPS Processor Components
  – Future Work
  – Conclusion
Demo
Hardware Verification Methodology

Verilog Design

DE2 Design

Verifying Compiler

ACL2 Model

Compiler

Testing & Inspection

English Spec, C Model and Test Suite

Manual Translation

Guided Proof

ACL2 Spec

SAT-Based Decision Procedure

SULFA Properties
TRIPS LSQ

• TRIPS Processor
  – Designed and built at University of Texas and IBM
    • Prototype next-generation processor
  – Multi-core, speculative, pipelined processor
    • 4 memory partitions, 16 ALUs
    • 256 speculative out-of-order instructions, partitioned into eight instruction blocks

• Load Store Queue (LSQ)
  – Queue for speculative loads and stores not ready for cache
  – Four LSQ tiles, one for each memory partition

• Verified LSQ internal communication protocol
Exception Mask Protocol

- Exception can occur at each tile
- Each tile stores a mask of known exceptions
- Mask sent up each cycle
- Eventually every exception is known by **Tile 0**
- Global flushes remove exceptions
Verification of Exception Protocol

• Compiled Verilog design into DE2
• Compiled DE2 into ACL2 model
  – proven equivalence
• Wrote single-tile exception model
• Specification:
  – **Safety.** Tile 0 reports a subset of the exceptions reported by the single-tile model
  – **Liveness.** Eventually every exception produced by the single-tile model is reported by Tile 0.
Exception Protocol Safety Property

- Tile 0 reports a subset of the exceptions reported by the single-tile model

(defthm specification-miss-exception-safety
  (implies
   (and (integerp tao)
        (<= 0 tao)
        (Tth-valid-inputsp tao input-list))
   (submaskp
    8
    (acl2v-udt_miss_ordering_exceptions
     *t0*
     (Tth-model-state tao input-list)
     (nth tao input-list))
    (spec-miss-exceptions
     (Tth-spec-state tao input-list)
     (nth tao input-list))))
Safety Invariant Properties

(defthm miss-order-inv-is-invariant-step
  (implies
   (and (inputs-goodp proof-st ins)
        (miss-order-inv proof-st))
   (miss-order-inv (update-proof-state proof-st ins)))
  :hints ("Goal" :external (sat nil sat::$sat))))

(defthm miss-order-inv-implies-thm
  (implies
   (and (miss-order-inv proof-st)
        (inputs-goodp proof-st ins)
        (submaskp 8
                   (acl2v-udt_miss_ordering_exceptions *t0*
                    (proof-st-dsn-state proof-st) ins)
                   (update-proof-st-0th-miss-mask *t0* proof-st ins)))
   :hints ("Goal" :external (sat nil sat::$sat))))
Liveness Property

- Eventually every exception produced by the single-tile model is reported by Tile 0
- ACL2 Specification
  - Prove theorem below
  - Use **defun-sk** definition on next slide
- Proof process same as before
  - Unable to prove invariant directly with SAT

(defthm specification-miss-exception-liveness
 (implies
 (and (integerp tao)
 (<= 0 tao))
 (eventually-1T-miss-subset-of-4T-P tao input-list))
Liveness Property Defun-sk

(defun-sk eventually-1T-miss-subset-of-4T-P
  (tao input-list)
  (exists
   (tao-prime)
   (and (integerp tao-prime)
        (<= tao tao-prime)
        (implies
         (Tth-valid-inputsp tao-prime input-list)
         (submaskp
          8
          (spec-miss-exceptions
           (Tth-spec-state tao input-list)
           (nth tao input-list))
          (bv-or 8
                 (recent-flushes tao tao-prime input-list)
                 (acl2v-udt_miss_ordering_exceptions
                  *t0*
                  (Tth-model-state tao-prime input-list)
                  (nth tao-prime input-list))))))))
Store Mask Protocol

- Each tile produces a mask of arrived stores
- Protocol more complex than exception protocol
  - Up to 256 entries in store mask
  - Store mask sent to both neighbors
- Specification & verification methodology similar to exception protocol
- Analysis
  - Problem size: ~1500 Boolean variables
  - 130 hours of human effort
  - Multi-hour proof
- Improvement over pure theorem proving
  - Counter examples especially helpful
Future Work

• More applications
  – Full LSQ design
  – n-tile circuit generator

• Performance improvements
  – Try the new BDD system

• Expand SULFA
  – Constrained functions
  – Limited arithmetic

• Add to ACL2 “waterfall”

• Verify proof engine
  – Theoretical issues: function body, proof of termination
  – Practical issues: complex code, large clause inputs

• Counter-example guided refinement
Conclusion

• Defined decidable subclass of ACL2 formulas
  – Includes primitives if, cons, car, cdr, consp, and equal
  – Can be extended with user-defined functions
• New hint for proving properties in this subclass
  – Fully automatic
  – Generates counter-examples to invalid properties
• Applying to TRIPS processor
  – Multi-core, pipelined, out-of-order processor
  – Combining SAT-based reasoning with pure theorem proving
  – Solid improvement over pure theorem proving
Backup Slides
Single-Tile Exception Model

- Wrote a single-tile model in ACL2
- The full mask of exceptions is generated each cycle

* This input has been modified: an exception is disabled if it occurs in an instruction that has already been flushed.
Single-Tile Exception Model

• Wrote a single-tile model in ACL2
• The full mask of exceptions is generated each cycle

* This input has been modified: an exception is disabled if it occurs in an instruction that has already been flushed.
Store Mask Protocol

- Each tile keeps a mask of arrived stores
- Used in completion detection & deferred load awakening
- Up to three stores are sent both upward and downward at the beginning of each cycle
- Eventually all arrived stores reach every tile
- A flush or a commit removes stores
Single-Tile Store Model

• Similar to single-tile exception mask

*This input has been modified: an exception is disabled if it occurs in an instruction that has already been flushed.*