Industrial Hardware and Software Verification with ACL2

Warren A. Hunt, Jr.[1], Matt Kaufmann[1], J Strother Moore[1], and Anna Slobodova[2]

[1] Department of Computer Science
University of Texas at Austin
and
[2] Centaur Technology, Inc.
Austin, TX.

April, 2016

Question: "How did ACL2 come to be used by industry?" — Philippa Gardner

Short Answer: It's an efficient programming language with a good prover, the user community is cohesive and largely industrial, and *the goal of the project is to make hardware/software verification practical*.

Introduction

ACL2: A Computational Logic for Applicative Common Lisp

Logic: first-order extended subset of applicative Common Lisp

Home page:
`www.cs.utexas.edu/users/moore/acl2`

Distribution: without charge, source-code form, 3-clause BSD license

Implementation: 87% of the 5.7 MB of source code is in ACL2; rest in "raw" Common Lisp; runs under six independent Common Lisp implementations on many hardware/OS platforms

Online Documentation: 68 MB

Lemma Library: `github.com/acl2/acl2`; 5,780 "certified books" contributed by many users; $\sim$ 123,000 theorems

The Origin Story

Computational Logic, Inc. (CLI) was founded in 1987 to spread verification technology to industry, initially using Nqthm.

ACL2 was started at CLI in 1989 to address inadequacies of Nqthm in industrial-scale projects like the CLI Verified Stack and the verification of MC68020 binary code.

Design Goals for ACL2, Kaumann and Moore, 1994:

> Foremost among those inadequacies is the fact that Nqthm's logic is an inefficient programming language. We now recognize that the efficiency of the logic as a programming language is of great importance because the models of microprocessors, operating systems, and languages . . . must be executed . . .

Initial Industrial Demonstrations

CLI was contracted to use ACL2 for:

- Motorola CAP DSP, 1993 – modeled hardware and microcode and verified an executable predicate to recognize all pipeline hazards

- AMD K5 Microprocessor FDIV operation, 1995 – verified IEEE 754 compliance of FDIV microcode, before fab.

In addition to stressing and improving the tool, these high-risk verification projects engendered a team-spirit and shared vision among CLI researchers using ACL2

Dispersion

CLI closed its doors 1997-99.

ACL2 users dispersed to CLI clients including AMD, Motorola, IBM, Rockwell-Collins

ACL2 was subsequently used in many experimental industrial projects, led by ex-CLI researchers

- AMD: All elementary floating point on Athlon – after running 100M test vectors comparing the ACL2 model to the AMD RTL simulator; Opteron and all AMD desktop machines
- Rockwell-Collins: silicon JVM chip, AAMP7 crypo-box, Greenhills OS
- IBM: Power 4 FDIV and SQRT; and a deep embedding of a hiearchical HDL

- UT ACL2 Group: Sun JVM class loader and byte-code verifier

- Boyer and Hunt developed an experimental version of hash cons and a prototype Common Lisp tool for bit-blasting ACL2 expressions (verified in 2010)

Integration into Design Workflow

In 2007, Centaur Technology, Inc., challenged Hunt to verify the VIA Nano floating-point adder design:

- handles single (32-bit), double (64-bit) and extended (80-bit) additions
- pipelined to deliver 4-results per cycle
- 33,700 lines of Verilog

- 680 modules
- 432,322 transistors
- part of the media unit with 1074 input signals (including 26 clocks) and 374 output signals

In about 9 months' work, coding in ACL2, Hunt and a grad student (Swords):

- improved the ACL2 HDL to handle the Centaur Verilog subset producing an executable semantic model called E
- wrote a translator from Verilog to E
- developed techniques to slice models to use bit-blasting and glue pieces together with proof

- executed E model of the media unit to test the adder and identify proper inputs to activate adder
- verified the adder – finding a bug in the 80-bit case

ACL2 at Centaur Today

ACL2 is an indispensable part of the Centaur design process

Centaur FV team consists of 3 full-time employees and a couple of interns

While its main focus is the validation of the RTL design, the FV team's wider goal is to provide design and verification support at various abstraction levels.

Centaur's current family of x86-based microprocessors is called VIA Eden

Centaur has an ACL2 specification of the Eden subset of the x86

Validated by routinely running millions of tests comparing ACL2 x86 to Intel, AMD, and Centaur hardware

The ACL2 tool-chain translates the entire Eden design (700,000+ lines of Verilog) into a formal object in a few minutes

The translated model is validated by running millions of tests against Cadence NC Verilog and Synopsys VCS Verilog simulators

Centaur's Verilog tool-chain is distributed in the ACL2 books and is used by Intel and Oracle

Two main applications:

- proving correctness of parts of the RTL design wrt behavioral spec
- proving correctness of microcode (typically no longer than several hundred lines)

All proofs are re-run nightly

"*Bugs introduced today are found tonight and fixed tomorrow.*"

Centaur uses ACL2 and the Verilog translation of the design to build custom tools including

- a linter which is routinely run several times per day and sends reports to logic designers
- an RTL design browser
- a reverse engineering tool, e.g., to extract the clock tree

- a tool to produce understandable reports about synthesized circuits by finding gate-to-RTL signal correspondences

All Centaur analysis tools are driven off the same ACL2 object representing the design

Analysis tools are written in ACL2

Other Ongoing Industrial Projects

The paper additionally lists some ongoing ACL2 projects at

- AMD (transaction protocols)
- Intel (Elliptic curve crypto)
- Kestrel Institute (Android apps)
- Oracle (floating point)
- Rockwell-Collins (LLVN)

Strengths

- fast execution
- proof/expressive power
- programmable/verifiable extensibility
- system programming capabilities
- proof automation and re-play
- trust tags (allowing calls to unverified external code)

Weaknesses (reported by industry)

- inefficient execution of some primitives
- inconvenient as a scripting language
- does not support visualization/graphics tools

Centaur uses Perl, Ruby, Makefile, C, HTML, Javascript, CSS, and a Common Lisp web server, all connected to ACL2 (when necessary and non-critical) via the underlying Common Lisp features and trust tags

Critical analysis tools are written in pure ACL2

Many critical tools (e.g., bit-blasting, Verilog transformations) are verified

Weaknesses (reported by industry)

- inefficient execution of some primitives
- inconvenient as a scripting language
- does not support visualization/graphics tools

Note: Industry's complaints about ACL2 rarely concern absence of strong typing, explicit quantifiers, partial functions, or higher order functions

Support for Industrial Projects

There have been over 1000 changes since Centaur started using ACL2 in May, 2009. Of those, these were requested by Centaur:

Changes to Existing Features	95
New Features	44
Heuristic and Efficiency Improvements	22
Bug Fixes	72
Changes at the System Level	18
Total due to Centaur	251

Why ACL2 is Successful in Industry

- that was the goal of the project
- efficient, executable logic/programming language with native verifier
- dual-use bit- and cycle-accurate models
- access to Common Lisp programming (via trust tags)
- automatic prover with "a human in the loop"

- rugged, well documented, free, open source form, many useful books, and a fairly unrestrictive license.
- coherent user community devoted to making mechanized verification practical
- industry needs help