Formal Specification and Verification of Floating-Point RTL with ACL2

David M. Russinoff
Arm Holdings

September 13, 2019
A verification methodology based on two components:

- **RTL**: An ACL2 library of definitions and lemmas pertaining to register-transfer logic, floating-point arithmetic, instruction specification and implementation
- **RAC**: An intermediate modeling language (subset of C++) and a translator to the ACL2 logic
IEEE Compliance

[Each elementary arithmetic operation] shall be performed as if it first produced an intermediate result correct to infinite precision and then rounded that result ...

\[
\begin{align*}
\text{Rational Operands} & \quad + \quad \times \quad \div \quad \sqrt{} \\
\text{Precise Result} & \quad \rightarrow \quad \text{Round} \\
\text{Rounded Result} & \quad \text{Encode} \\
\text{Decode} & \quad \text{Bit Vector Inputs} \\
\text{Bit Vector Output}
\end{align*}
\]
THE ACL2 RTL LIBRARY

(1) Register-transfer logic: bit vectors and logical operations
(2) Floating-point arithmetic: FP decomposition, formats, rounding
(3) Behavioral specifications of elementary arithmetic operations
(4) Implementation: algorithms and techniques
(1) Register-Transfer Logic

- **Bit slice and bit extraction operators:**
  \[ x[i : j] = \lfloor (x \mod 2^{i+1})/2^j \rfloor \]
  \[ x[i] = x[i : i] \]

- **Logical operations:**
  \[ \sim x = -x - 1 \]
  \[ x \mid y = \begin{cases} y & \text{if } x = 0 \text{ or } x = y \\ x & \text{if } y = 0 \\ 2(\lfloor \frac{x}{2} \rfloor \lfloor \frac{y}{2} \rfloor) + (x \mod 2) \mid (y \mod 2) & \text{otherwise} \end{cases} \]

- **Derived bitwise characterizations:**
  \[ \sim x[i] = 1 - x[i] \]
  \[ (x \mid y)[i] = x[i] \mid y[i] \]
(2) **Floating-Point Arithmetic**

- FP decomposition: \( x = sgn(x) \cdot sig(x) \cdot 2^{\text{expo}(x)} \)
- Exactness: \( x \) is \( n \)-exact \( \iff \) \( 2^{n-1}sig(x) \in \mathbb{Z} \)
- Rounding according to various modes:

\[
RTZ(x, n) = sgn(x) \lfloor 2^{n-1}sig(x) \rfloor 2^{\text{expo}(x)-n+1}
\]

\[
RTO(x, n) = \begin{cases} 
  x & \text{if } x \text{ (n-1)-exact} \\
  RTZ(x, n-1) + sgn(x)2^{\text{expo}(x)+1-n} & \text{otherwise}
\end{cases}
\]
Variations in architectural behavior, especially regarding exceptional conditions (e.g., underflow, denormal operands, exception precedence and interaction) necessitate separate specifications for the ISAs of interest:

- x87
- SSE
- Arm
4) Algorithms and Optimizations

- Addition: Carry-look-ahead adders, leading zero anticipation, trailing zero anticipation
- Multiplication: Booth encoding schemes
- Division and square root: SRT algorithms, FMA-based division
REPRESENTATION OF RTL IN ACL2

- AMD: Verilog-ACL2 translator generates an ACL2 function corresponding to each RTL signal
- Centaur: Verilog module converted to netlist of S-expressions and executed by ACL2 hardware interpreter
- Arm: Intermediate C model is derived from RTL, checked by SLEC, and translated to ACL2
A common industrial practice is to check equivalence between an RTL design and a trusted high-level C++ model with a commercial tool. This approach suffers from two deficiencies:

- Legacy model is trusted because it has been used/tested extensively, but it is inadequate as a specification and has never been formally verified itself.
- Micro-architectural gap between C model and RTL pushes SLEC technology to its limits.

  - Limited opportunities for proof decomposition
  - Elaborate proof scripts introduce possible source of error
  - Complex SP operations take all day to check
  - DP ops (FMA, FDIV, FSQRT) fail to converge
**Restricted Algorithmic C (RAC)**

- A primitive subset of C augmented by the register class templates of Algorithmic C (Mentor Graphics)
- A design-specific RAC model, which performs the essential computations of the design while eliminating implementation details, may be efficiently checked against RTL (SLEC, Hector)
- The RAC translator generates a compact ACL2 model that may be mechanically verified against an architectural specification
RAC Features

- Numerical data types: bool, uint, int (no pointers)
- Composite types: arrays, structs, enums
- Standard control constructs: if, for, switch, return (with certain restrictions)
- Functions (value parameters only)
- Arbitrary width integer and fixed point register class templates (Algorithmic C)
- Standard library class templates: array, tuple (facilitate parameter passing)
Translation to ACL2 is a two-step process:

- C++ (Flex/Bison) parser produces an internal representation
  - C syntax $\rightarrow$ S-expressions
  - Variable types $\rightarrow$ Explicit conversions

- ACL2 program generates ACL2 functions
  - Assignment sequences $\rightarrow$ Nested bindings
  - Iteration $\rightarrow$ Recursion

The parser also produces a pseudocode version of the model (C/Verilog syntax) suitable for design documentation.
Case Study: FP Divider of an Arm Processor

- Radix-4 SRT FDIV and FSQRT (HP, SP, DP)
- 27 DP iterations, 3/cycle for FDIV, 2/cycle for FSQRT
- 259 KB Verilog $\rightarrow$ 35 KB RAC $\rightarrow$ 42 KB ACL2
- C Model and RTL produce identical remainders and quotients at each iteration
- SLEC execution time linear in operand width, 87 minutes for DP FSQRT
- ACL2 proof script consists of 1600 lemmas
- Level of effort (modeling + verification): 3 man-months
The RTL library and the RAC parser and translator reside in the community books directory of the ACL2 repository (http://www.cs.utexas.edu/users/moore/acl2):

▶ books/rtl/README
▶ books/projects/rac/README

Documentation: