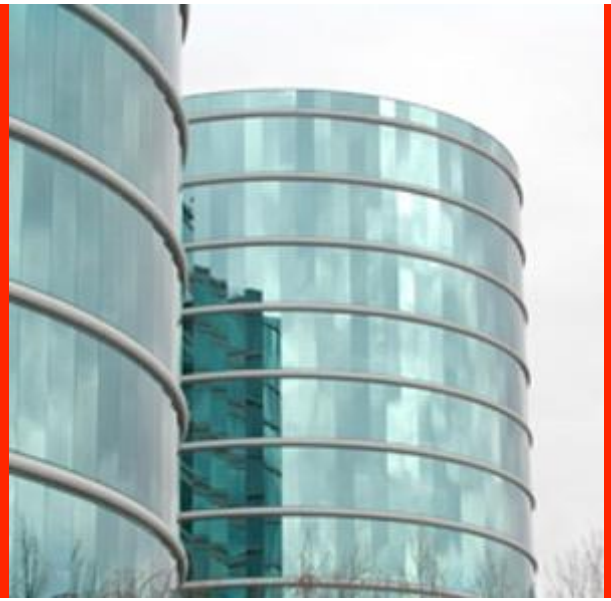


ORACLE®

A Brief Introduction to Oracle's Use of ACL2 in Verifying Floating-Point and Integer Arithmetic

David L. Rager, Jo Ebergen, Austin Lee, Dmitry Nadezhin, Ben Selfridge, Cuong K. Chau

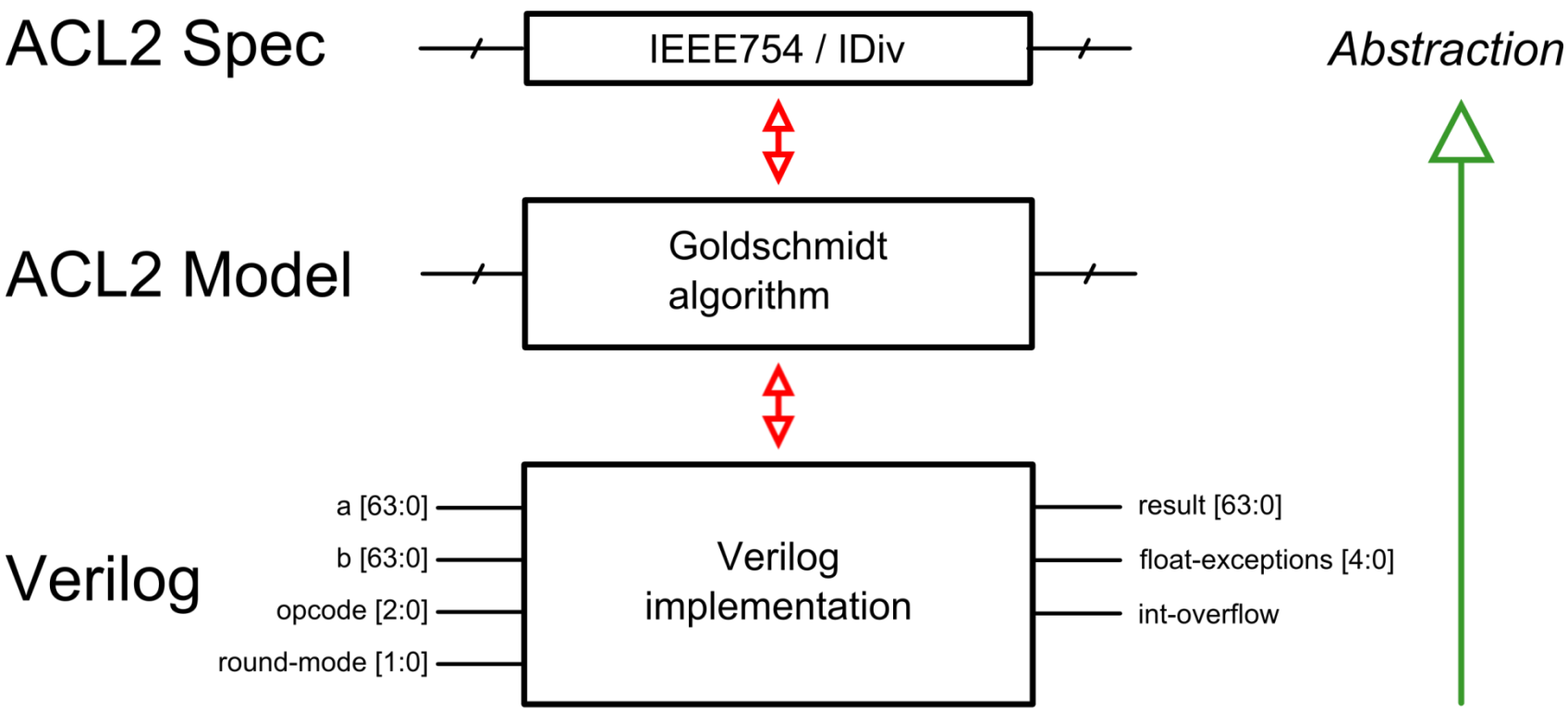
October 1, 2015

Goal

- Verify data-path for:
  - 32/64-bit floating-point division and square root
    - fdivd
    - fdivs
    - fsqrtd
    - fsqrts
  - 32/64-bit integer divide
    - udivx
    - sdivx
    - udiv
    - sdiv

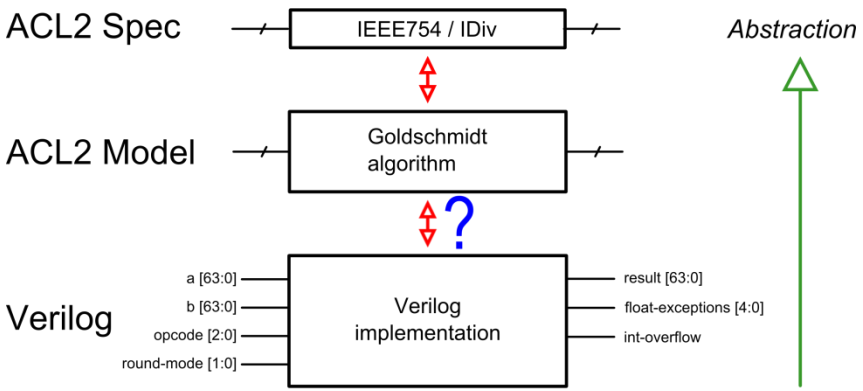
    This work is similar in spirit to work done by AMD, Centaur, and others.

Breaking Down the Problem



Outline

- Goal
- *Algorithm extraction*
- Algorithm verification
- Verifying our process
- Conclude



Algorithm Extraction (part A)

- Parsed the circuit into combinations of low-level logical primitives
- Read design using VL2014+Esim (~535 books)
- Abstracted low-level logical primitives via GL into medium-level primitives

```
(defstv add16-test-vector

  :mod *add16*
  :inputs '((“clk" 0 ~)
            (“abus" a)
            ("bbus" b))
  :outputs '(("out" _ _ out)))
```

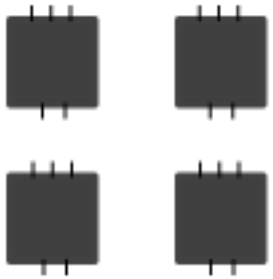
```
(def-gl-thm add16-adds
  ...
  :concl (equal (cdr (assoc ‘out (stv-run *add16* (list (cons ‘a a)
                                                        (cons ‘b b))))))
                (mod (+ a b) *2^16*)) ...)
```

Algorithm Extraction (part B)

- Converted medium-level primitives into higher-level concepts like + and *
- Goldschmidt algorithm could be expressed in these high-level mathematical primitives
- But…

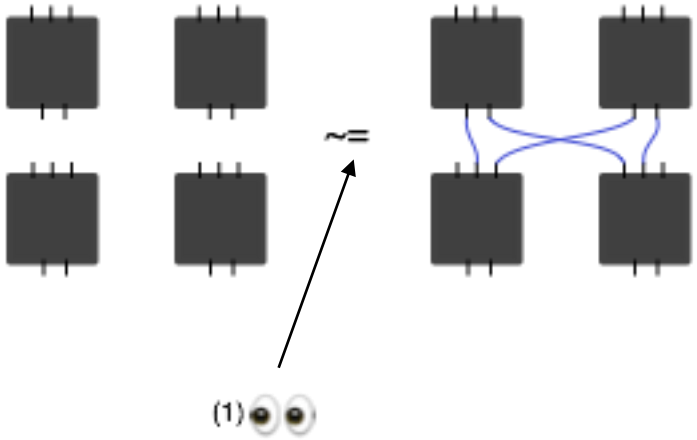
Breaking Up Is Hard To Do

- Decompose circuit into appropriately-sized blocks
- Start with modules already “black-boxed” via GL
  - For example, a tree of carry-save adders (CSAs)



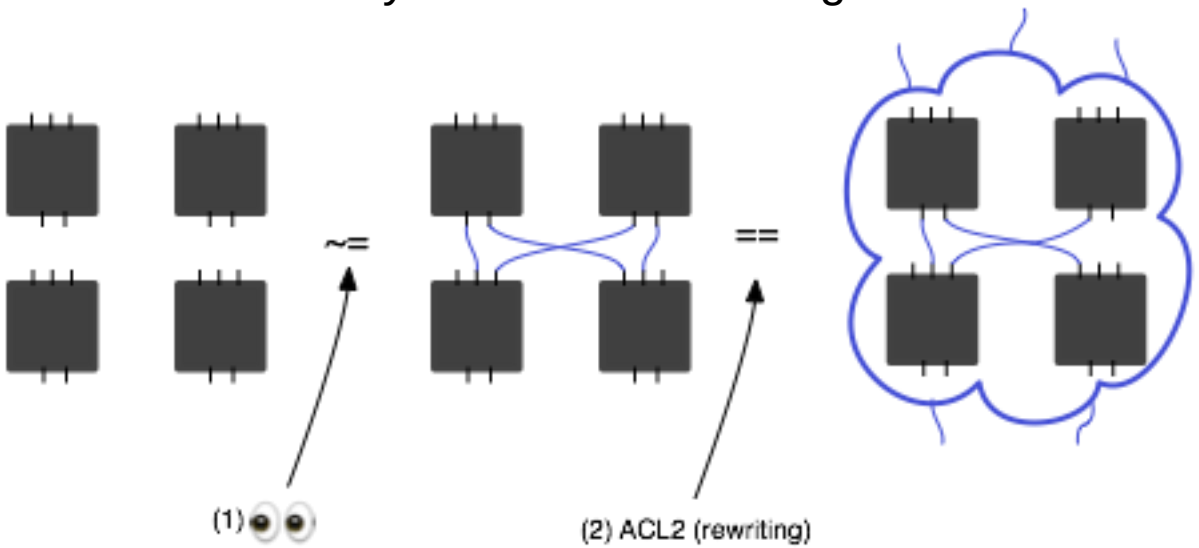
Breaking Up Is Hard To Do

- Decompose circuit into appropriately-sized blocks
- (1) Create ACL2 version of the interconnect
  - E.g, the wires that connect the CSAs are connected in a particular way



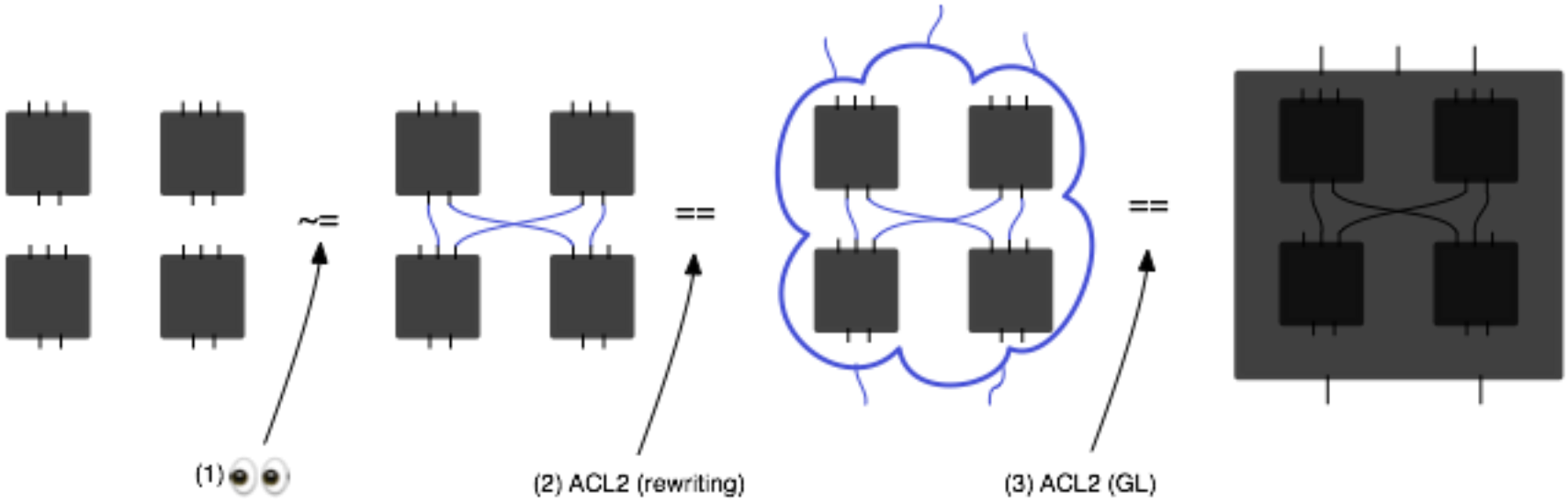
Breaking Up Is Hard To Do

- Decompose circuit into appropriately-sized blocks
- (2) Prove that connecting the modules in that way meets a specification
  - E.g, assuming the wires are connected that way, this property holds: sum+carry*2 = a+b+c+d+e+f+g+h



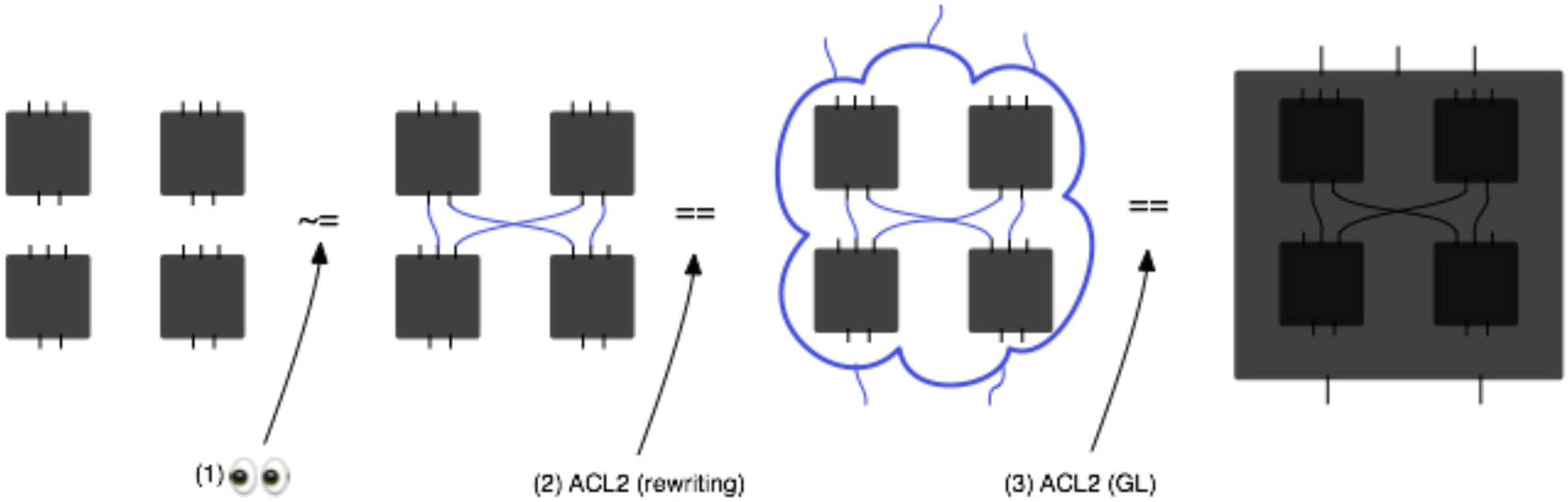
Breaking Up Is Hard To Do

- Decompose circuit into appropriately-sized blocks
- Prove that the ACL2 interconnect is the same as the Verilog interconnect
  - E.g, that the Verilog wires really do connect the CSA's that way!



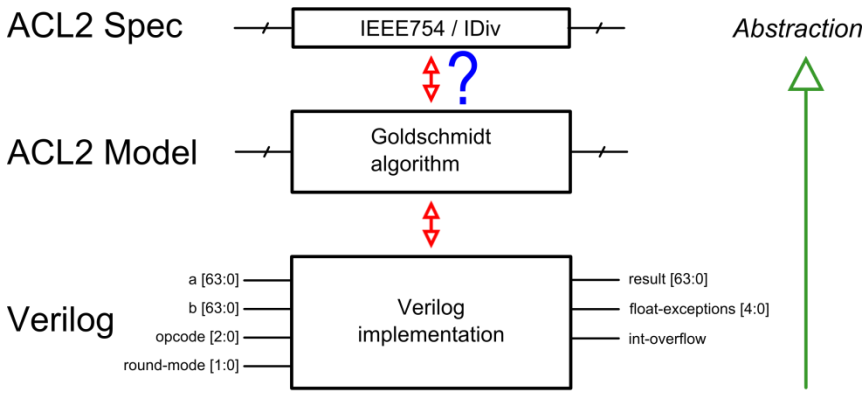
Breaking Up Is Hard To Do

- Decompose circuit into appropriately-sized blocks
- Prove that the ACL2 interconnect is the same as the Verilog interconnect
  - Proof (3) via GL using Esim doesn't scale, use SV in the future

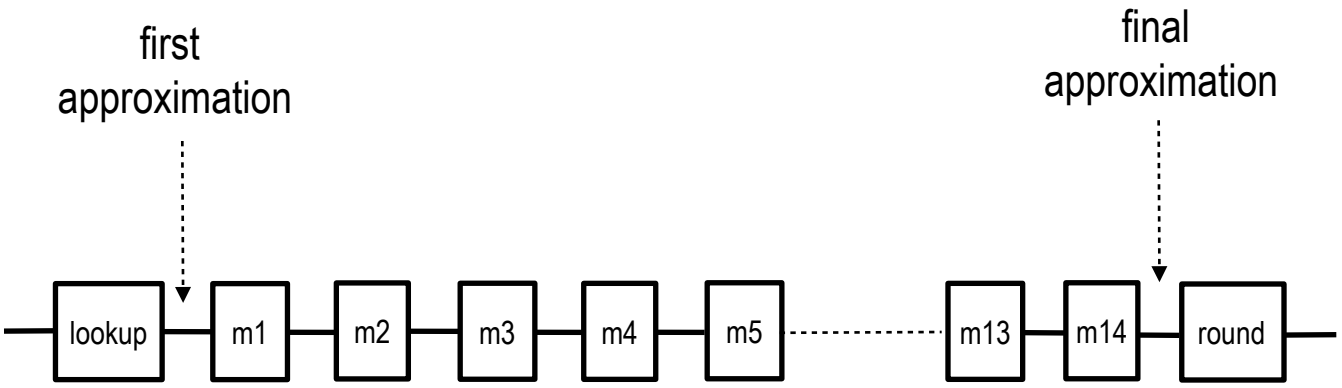


Outline

- Goal
- Algorithm extraction
- *Algorithm verification*
- Verifying our process
- Conclude



A Very Simple Model* of the Goldschmidt Division Algorithms



- Most margins of error come from:
  - Initial lookup table
  - Truncation of intermediate multiplication results
- Golden question: *Is the final approximation accurate enough to yield an IEEE754 answer after rounding is applied?*

* this is an intentionally obfuscated model – it may look confusing to those familiar with optimized Goldschmidt implementations

IEEE Specification

- An IEEE754 specification for our purposes should:
  - Specify add, subtract, multiply, divide, fused-multiply-add, and square root operations
  - Specify denormals
  - Specify exceptions
  - Permit us to propagate NaN payloads in a way consistent with Sparc
  - Capture the effects of four rounding modes
  - Deal with over/underflow

Outline

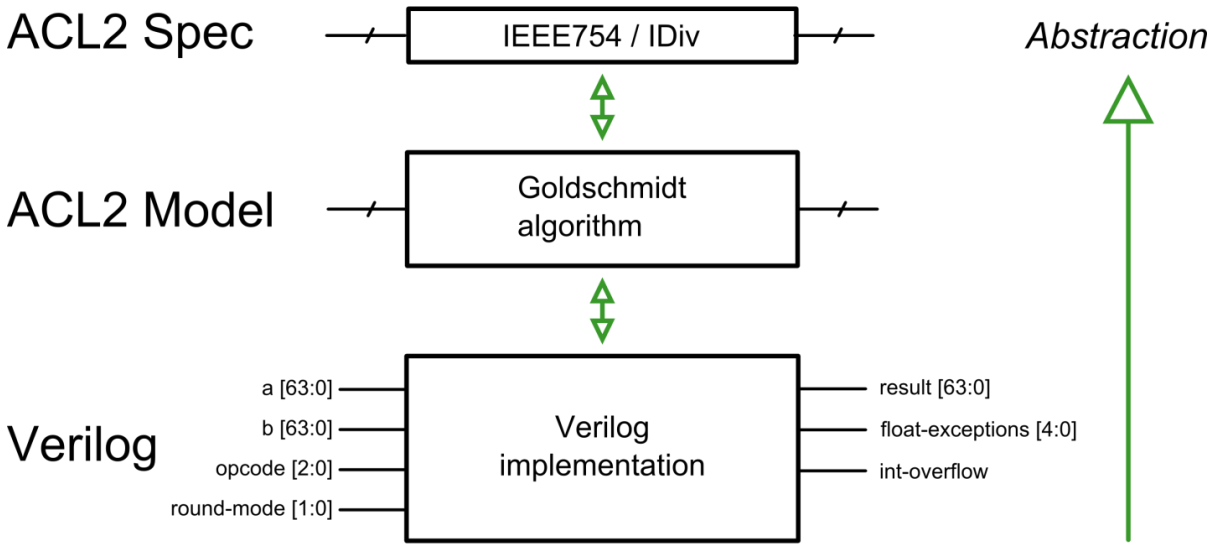
- Goal
- Algorithm extraction
- Algorithm verification
- *Verifying our process*
- Conclude

Verifying Our Process

- Exercise: designer introduced 3 independent errors
  - All errors broke our proofs…
  - ...*exactly where the proofs were supposed to break*.
  - Not really surprising
- Validated specifications and models with concrete test vectors
- We run nightly regressions of all proofs
- Uses actual Verilog implementation
  - Guards against introduction of errors late in design

Conclusion

- Capable tool chain
- Cannot achieve necessary coverage without such rigorous analysis
- No bugs in design
- Thorough analysis yields optimizations
  - e.g., 50% and 75% reduction in lookup tables




ORACLE®