Centaur Technology Media Unit Verification

Warren A. Hunt, Jr. and Sol O. Swords

June, 2009

Computer Sciences Department
University of Texas
1 University Way, M/S C0500
Austin, TX 78712-0233

E-mail:
{hunt,sswords}@cs.utexas.edu
TEL: +1 512 471 {9748,9744}
FAX: +1 512 471 8885

Centaur Technology, Inc.
7600-C N. Capital of Texas Hwy
Suite 300
Austin, Texas 78731

E-mail:
{hunt,sswords}@centtech.com
TEL: +1 512 418 {5797,5751}
FAX: +1 512 794 0717

Introduction

We have verified more than 50 media-unit (adds, subs, compares, converts, logicals, shuffles, blend, insert, extract, min-max) instructions from Centaur's 64-bit, X86-compatible microprocessor.

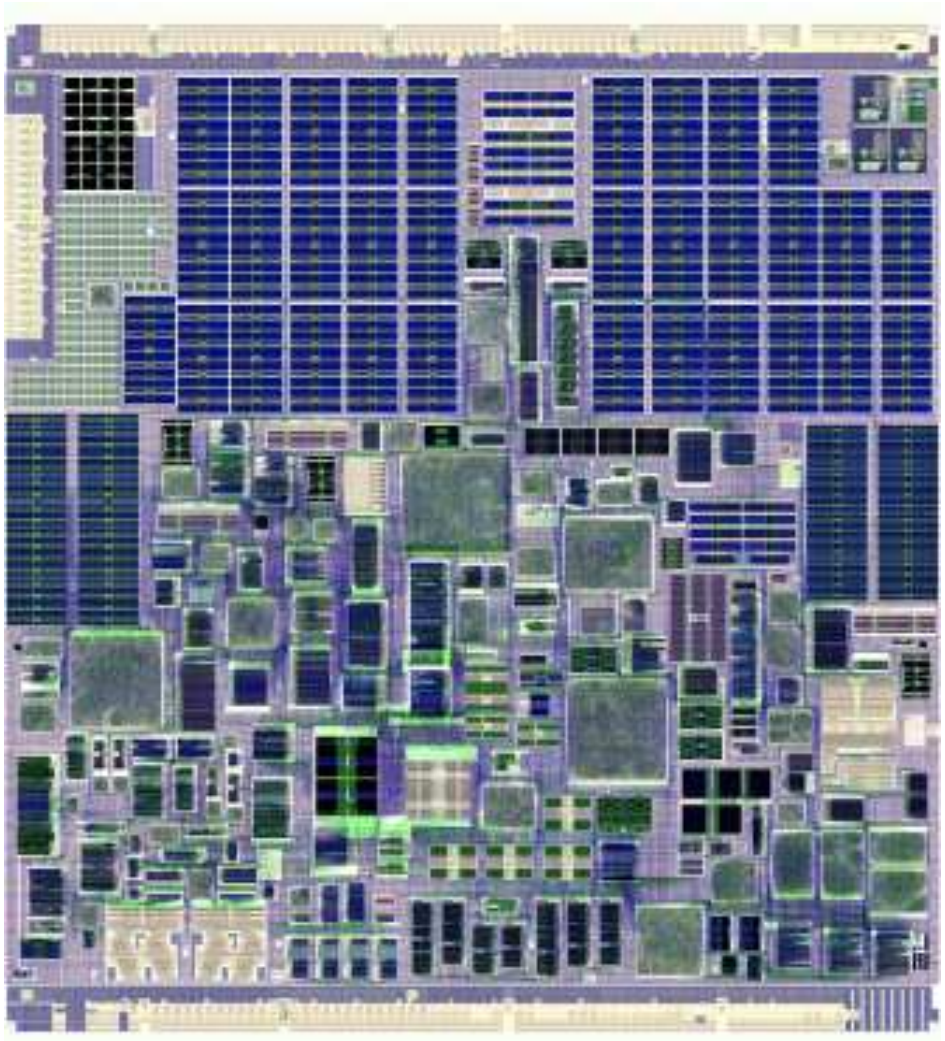
- Media unit implements over 100 X86 SSE and X87 instructions.
- Unit can add/subtract four pairs of floating-point numbers every clock cycle with an industry-leading two-cycle latency.

For our verifications, we use a combination of AIG- and BDD-based symbolic simulation, case splitting, and theorem proving.

- We create a theorem for each instruction to be verified.
- We use ACL2 to mechanically verify each proposed theorem.

Here, we discuss our verification of Centaur Nano^{TM} floating-point addition/subtraction instructions – the Nano^{TM} is used by Dell, HP, OLPC, and Samsung.

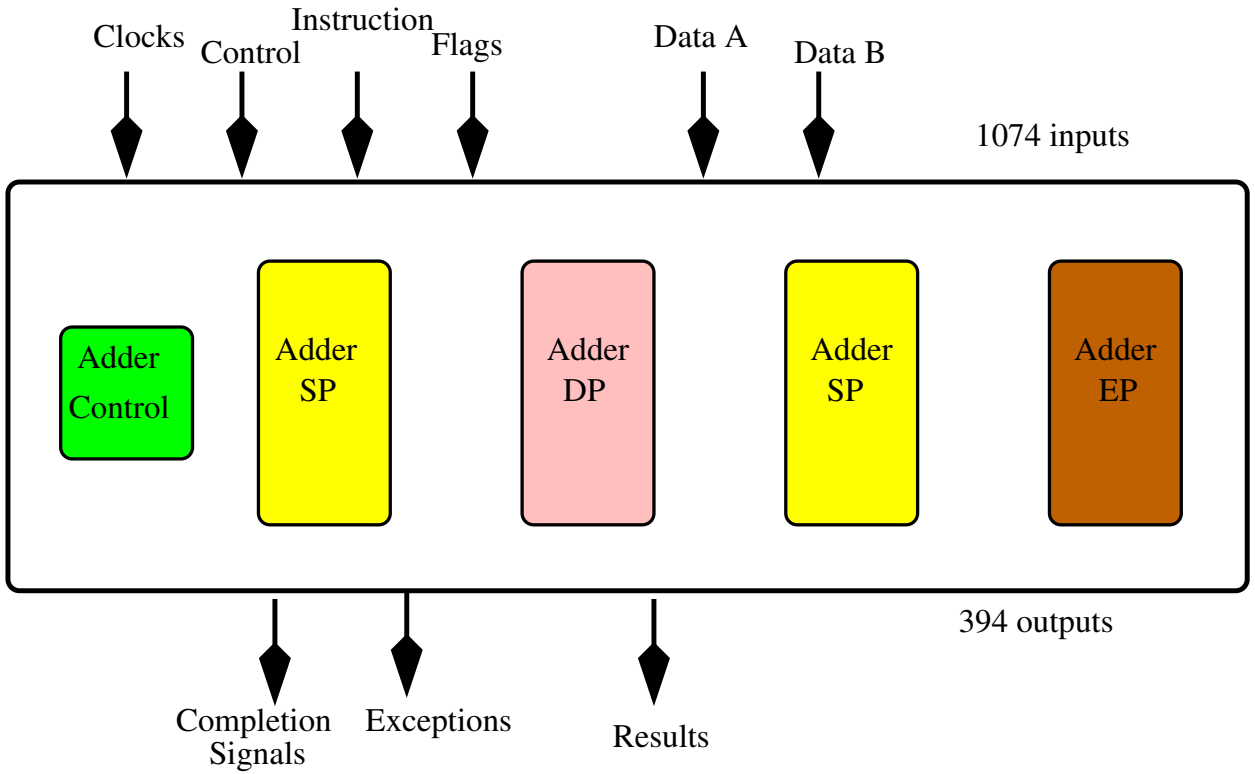
Centaur Nano^{TM} X86-64 Microprocessor



Contemporary Example

- Full X86-64 design including VMX
- 65-nanometer design of 94.5M transistors
- AES, DES, SHA, and random-number generator hardware
- Built-in security processor
- Runs 40 operating systems and four VMs

Centaur Nano^{TM} Media Unit – FADD



- 33,700 line Verilog description of 680 modules
- Modules represent 432,322 transistors
- Unit has 374 outputs and 1074 inputs (26 clocks)
- Implements over 100 media instructions
- Two-cycle-latency for floating-point additions/subtractions

Toolflow

We begin, by translating Nano's Verilog specification into our formally-defined, **E**-language HDL.

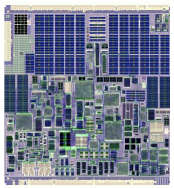
- Verilog is simplified into *single-assignment* form.
- Create environment suitable for media unit verification.
- We extract its *equation* by symbolic simulation.
- We specialize this equation to the instruction of interest.
- We then, as appropriate, convert this equation into BDDs.

The specification is written in ACL2.

- Integer operations are used to specify media-unit instructions.
- Such operations are symbolically simulated and specialized.
- These specification are proven to implement floating-point operations.

Finally, the results of both paths are compared.

The Centaur Media-Unit, Verification Tool Flow



Verilog Files

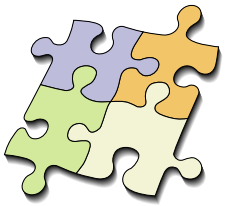
fadd EMOD module

EMOD Symbolic Simulator

fadd AIG function

Specialize

Per-instruction AIGs



Case-splitting, Parametrization

AIG2BDD

Instruction Spec

c=a+b

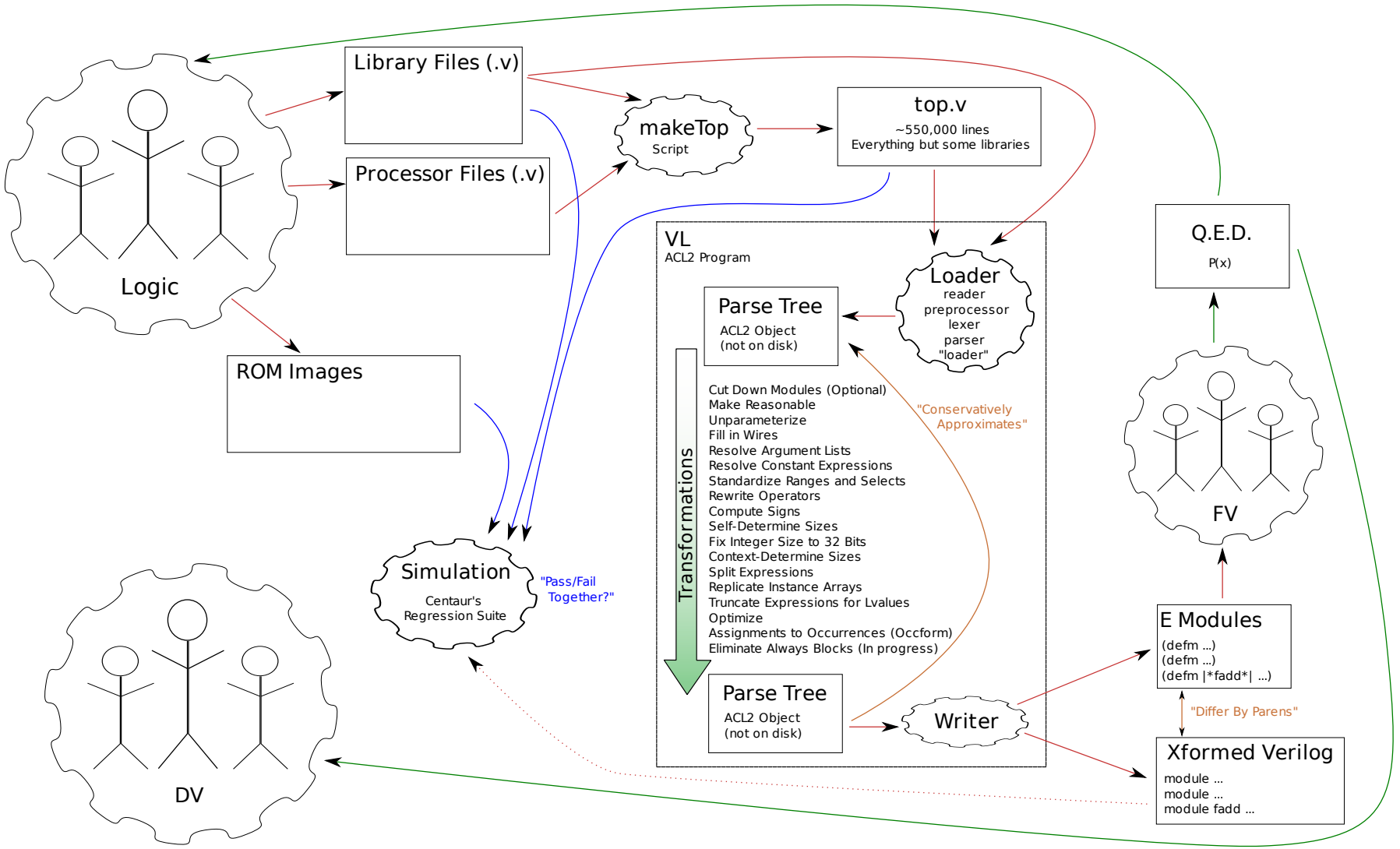
Symbolic Spec

Spec Output BDDs

=?

Hardware Output BDDs

The Verilog-to-E Translator (Jared Davis)



E-Language Features

The **E** language is deeply embedded in ACL2, and it is:

- hierarchical, and
- occurrence-oriented.

We use the **E** language much like a database; it includes:

- HDL descriptions
- Hierarchical state representation
- Signal sense and direction
- Clock discipline
- Properties
- Annotations

E-language has multiple symbolic simulators

- BDD and AIG (both two- and four-valued) simulators
- Symbolic information-flow simulator
- Delay estimator

E-Language Example

```
(defm *simple-ff-latch*
  `(:i (clk a)
    :o (o)
    :s (l- l+)
    :c (clk)
    :cd ((t) (nil))
   :occs
    ((:u l+ :o (n n~) :op ,*latch+*  :i (clk a))
     (:u o1 :o (clk~) :op ,*not1*    :i (clk))
     (:u l- :o (o o~) :op ,*latch+*  :i (clk~ n)))))
```

Simple two-latch, flip-flop

- Interface: **:i**, **:o**, and **:s** fields.
- Clock: **:c** and **:cd** fields.
- Occurrences are in a list, but treated as a set
- Multi-phase and gated clocking supported (and used by Centaur)

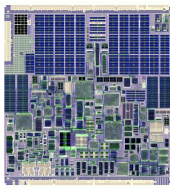
Symbolic Simulation in ACL2

We have created developed a verified framework for ACL2 that provides a means for symbolic simulation.

- Defined functions can be mechanically generalized.
- Such generalized functions, given finite sets, can be symbolically executed.
- Our framework allows the results of symbolic simulation of ACL2 functions to be used as a part of a proof.

Our work provides a symbolic-simulation capability for the entire ACL2 logic.

The Centaur Media-Unit, Verification Tool Flow



Verilog Files

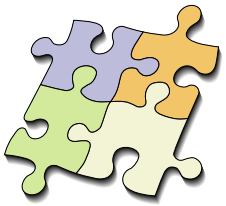
fadd EMOD module

EMOD Symbolic Simulator

fadd AIG function

Specialize

Per-instruction AIGs



Case-splitting, Parametrization

AIG2BDD

Instruction Spec

c=a+b

Symbolic Spec

Spec Output BDDs

=?

Hardware Output BDDs

Symbolic Simulation of the Media Unit

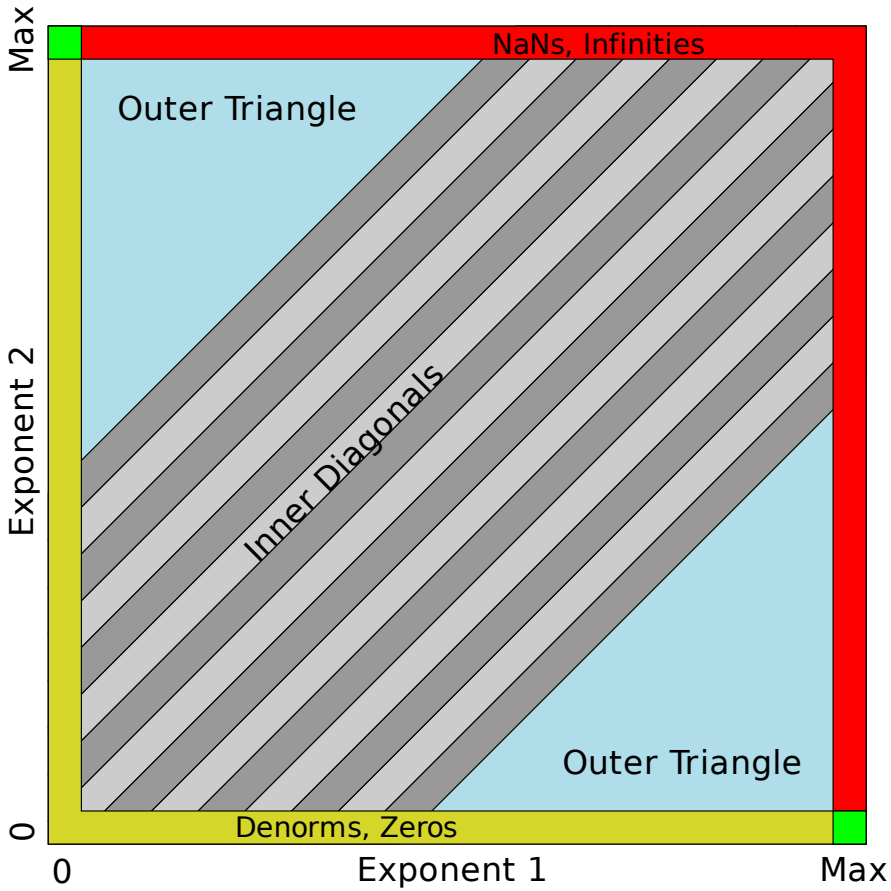
Using the **E**-language model, we perform a four-valued, AIG-based symbolic simulation of entire design for eight half-cycles.

- AIGs specialized for the instruction under investigation
- AIGs are converted to BDDs
  - For some instructions, a property may be too big to verify directly, so case splitting employed
  - For each case, BDD approximated until exact
  - For each case, compared to symbolic simulation of specification
- Cases are shown to be exhaustive

The Centaur Media-Unit, Case-Splitting Approach

For floating-point add/subtract, problem is too big to verify all at once.

- Case split by exponent differences
- Separately, account for special cases (e.g., NaNs, Infinity)
- For each case, generate symbolic inputs that cover exactly the specified set of inputs
  - BDDs are parametrized
  - Approach used for all FP sizes



Centaur Media-Unit, Verification

We attempted to verify single, double, and extended precision addition/subtraction operations.

- Single precision (32-bit) results and flags OK.
- Double precision (64-bit) results and flags OK.
- Extended precision (80-bit) results had an error.
  - Exactly one pair of numbers returned an incorrect answer
  - Sort of like a *perfect storm*; a 64-bit cancellation
  - Answer returned was twice as big as it should have been.

A fix was developed, and this bug has been eliminated. We have checked the correctness of the new design – it took less than an hour.

Robert Krug proved that our Boolean-based adder/subtracter specification is correct.

Conclusion

ACL2 is in everyday commercial use at Centaur Technology.

- Each night, entire design is translated
  - 550,000 lines of Verilog translated to **E**
  - Unable to translate some modules – working to finish translation
- New ACL2 executable containing all **E**-based modules is built each day.
  - Entire translation and build time about 30 minutes
  - Human verifiers get newest design version each morning
- Each night we recheck our proofs on the new model

Extending ACL2:

- by deeply embedding the **E** HDL,
- with AIG and BDD algorithms, which we mechanically verified, and
- by providing generalized symbolic simulation of all ACL2 functions,

it is possible to use a theorem prover to support an industrial hardware verification flow.