Combining SMT with Theorem Proving for AMS Verification

The best of both worlds

Yan Peng & Mark Greenstreet

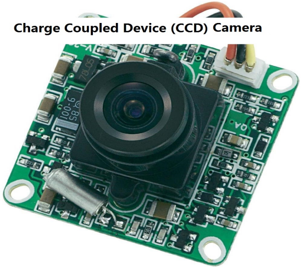
University of British Columbia
Vancouver, BC

March 30, 2015

Outline

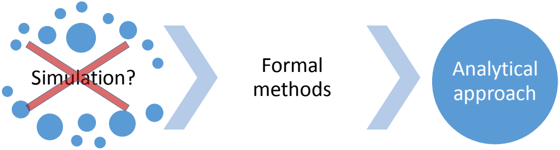
- AMS verification
  - AMS designs are ubiquitous
  - Motivation
  - Contributions
- The best of both worlds
- Integrating SMT with theorem proving
- Proving global convergence for a digital PLL
- Conclusion

AMS designs are ubiquitous



Motivation

- Different design models and methods, thus more complex to model.
- Time-scales vary widely from sub-picosecond to milliseconds or seconds, makes it harder to simulate.
- Abstractions on time-scales can hide bugs in the implementation.



- Circuits are intended to be correct
- Verify the intuitive argument

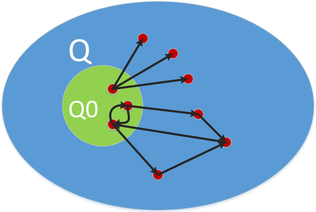
Contributions

- How to make use of the arithmetic decision procedures of an SMT solver for verifying properties of physical systems.
- The first integration of an SMT solver into the ACL2 theorem prover.
- First use of SMT solver in a theorem prover with emphasis on real-arithmetic.
- An software architecture for integrating a SMT solver with a theorem prover that addresses many technical challenges.
- A reusable recurrence model for a state-of-the art digital PLL.
- A proof of global convergence for the digital PLL.

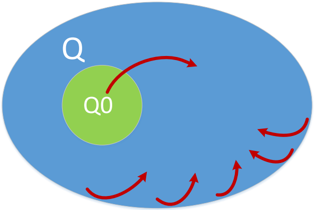
Outline

- Characterize AMS verification problems
- The best of both worlds
  - Formally characterize AMS verification problem
  - SMT and theorem proving
  - Simple example: prove sum of geometric series theorem
  - Simple example: prove given polynomial inequalities have no solution
- Integrating SMT with theorem proving
- Proving global convergence for a digital PLL
- Conclusion

Formally characterize AMS verification problem



$\forall s(0) \in Q_0 \forall i \geq 0.\ s(i) \in Q$



$\forall x(0) \in Q_0 \forall t \geq 0.\ x(t) \in Q$

Digital	Analog	AMS
$s(i+1) = next(s(i), in(i))$	$\frac{dx}{dt} = f(x, in, u)$	$\begin{aligned} \frac{dx}{dt} &= f_q(x) \\ q(i+1) &= d(q(i), th(x)) \end{aligned}$

Two features in formal model of AMS designs:

- Large non-linear arithmetic formulas
- Properties for sequences of states

SMT&Theorem proving

	SMT	**Theorem proving**
What	Satisfiability Modulo Theory	Computer aided theorem proving
Strength	Powerful (non)linear arithmetic solver and others	1. Systematic proof management 2. Induction proofs
Weakness	1. Lack of induction 2. Lemmas don't connect	Manual and tedious proofs
Tool	Z3[DMB08]	ACL2[KM97]

Geometric series theorem

Theorem (Geometric Sum)

Suppose $r \in \mathbb{R}$, $n \in \mathbb{N}$, $r > 0$ and $r \neq 1$. Then,

$$\sum_{i=0}^{n} r^i = \frac{1 - r^{n+1}}{1 - r}$$

Setup	LOC	# of theorems	runtime(s)	code time
raw Z3(can't)	-	-	-	-
raw ACL2(proved)	169	19	0.14	2 days
arithmetic-5(proved)	29	1	0.15	10 min
ACL2 & Z3(proved)	72	2	0.06	20 min

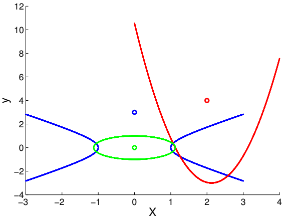
Polynomial inequalities

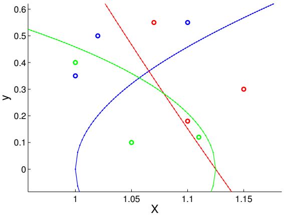
Theorem (Polynomial inequality)

Suppose $x \in \mathbb{R}$ and $y \in \mathbb{R}$, then

$$\begin{aligned} 1.125x^2 + y^2 &\leq 1 \\ x^2 - y^2 &\leq 1 \\ 3(x - 2.125)^2 - 3 &\leq y \end{aligned}$$

does not have a solution.





Polynomial inequalities

Theorem (Polynomial inequality)

Suppose $x \in \mathbb{R}$ and $y \in \mathbb{R}$, then

$$\begin{aligned} 1.125x^2 + y^2 &\leq 1 \\ x^2 - y^2 &\leq 1 \\ 3(x - 2.125)^2 - 3 &\leq y \end{aligned}$$

does not have a solution.

Setup	LOC	# of theorems	runtime(s)	code time
raw Z3(proved)	27	1	0.0004	10 min
raw ACL2(failed)	40	-	-	10 min
arithmetic-5(failed)	41	-	-	10 min
ACL2 & Z3(proved)	59	1	0.02	10 min

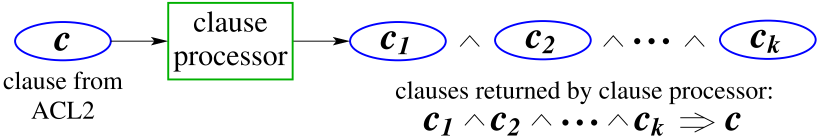
Best of both worlds - Summary

- AMS design verification requires huge amount of arithmetic reasoning and reasoning about sequences which requires induction.
- SMT and theorem proving are complimentary to each other:
  - SMT - Excellent performance in linear and non-linear arithmetic reasoning.
  - Theorem proving - Strong support for induction and systematical model & proof management.
- Small experiments show how one can benefit from combing them.

Outline

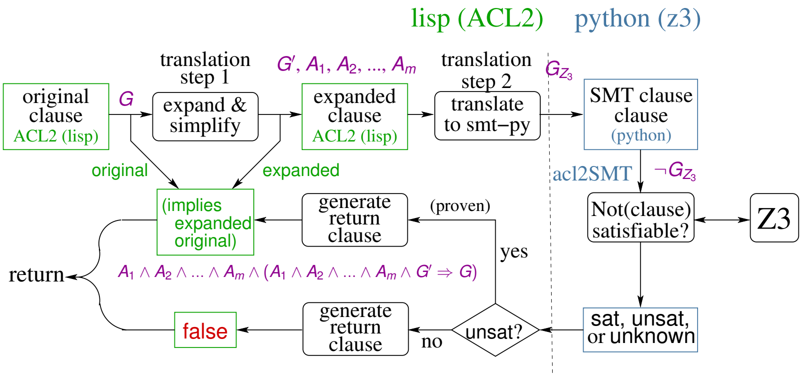
- Characterize AMS verification problems
- The best of both worlds
- Integrating SMT with theorem proving
  - Architecture
  - Technical issues
  - What's trusted?
- Proving global convergence for a digital PLL
- Conclusion

Starting from a clause processor

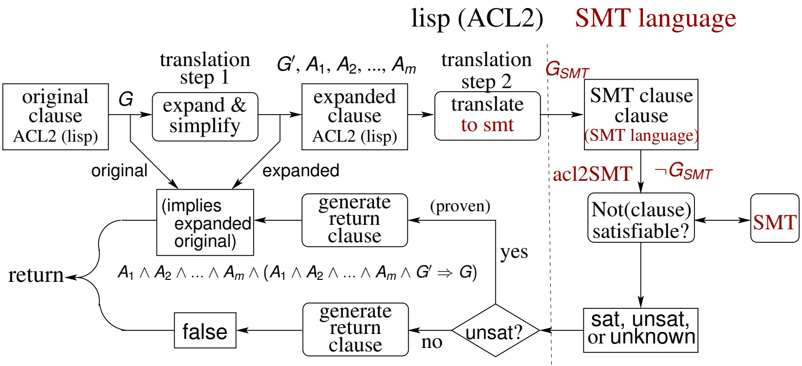


- A clause processor takes a goal and decomposes it into a conjunction of subgoals. Each subgoal is a called a clause.
- ACL2 supports two kinds of clause processors: verified and trusted.
  - verified - the correctness of the clause processor is proven within ACL2.
  - trusted - the results of the clause processor are accepted without proof.
- We integrate Z3 into ACL2 as a trusted clause processor.

Architecture

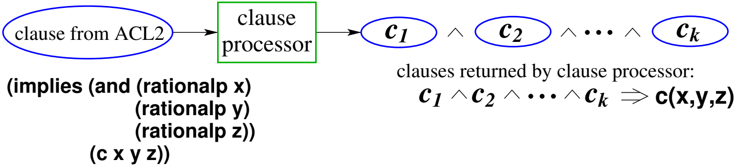


Architecture

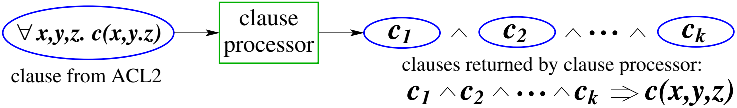


Technical issues

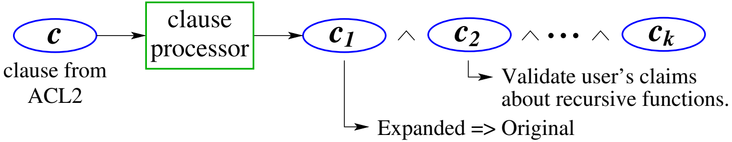
- typed vs. untyped



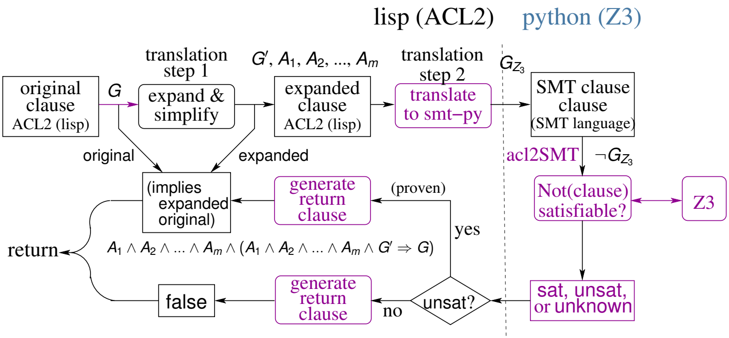
- reals vs. rationals



- user defined functions



What's trusted?



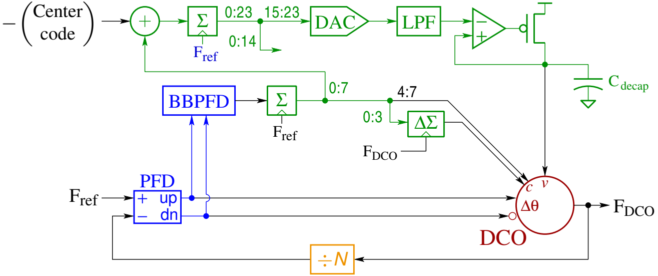
	translation	others	expansion & simplification
LOC(fraction)	656(39%)	453(27%)	584(34%)

- Translation and connection code are straight forward and easy to check.

Outline

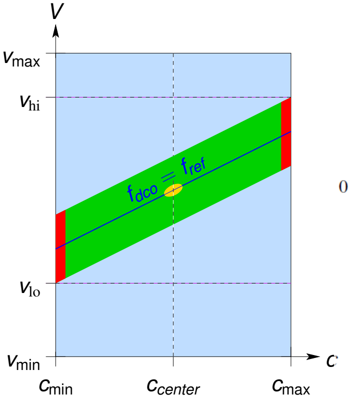
- Characterize AMS verification problems
- The best of both worlds
- Integrating SMT with theorem proving
- Proving global convergence for a digital PLL
  - The digital phase-locked loop
  - Modeling the digital PLL
  - Prove global convergence
- Conclusion

A state-of-the-art Digital PLL (from CICC 2010) [CNA10]

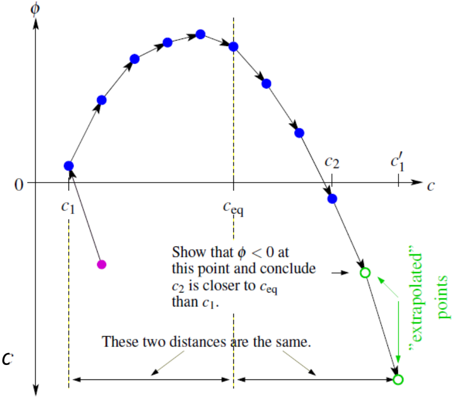


- A *PLL* outputs a signal with a frequency that's N times of the input signal. The output should also aligns the input in phase.
- Three state variables:
  capacitance setting (digital), supply voltage (linear), phase correction (time-difference of digital transitions).
- This is a typical AMS design.

The proof



V
v_{max}
v_{hi}
v_{lo}
v_{min}
$f_{dco} = f_{ref}$
c_{min}
c_{center}
c_{max}
c



ϕ
0
c_1
c_{eq}
c_2
c_1'
c
Show that $\phi < 0$ at this point and conclude c_2 is closer to c_{eq} than c_1.
These two distances are the same.
"extrapolated" points

Some statistics

- 13 page long hand-written proof.
- 75 lemmas, 10 of which were discharged using the SMT solver.
- Of those ten, one was the key, polynomial inequality from the manual proof.
- ACL2 completes the proof in a few minutes running on a laptop computer.
- We found one error in the process of transcribing the hand-written proof to ACL2.

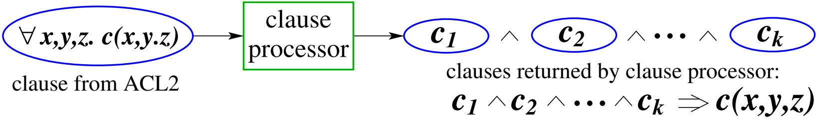
Conclusion

- We build a sound and extensible integration of an SMT solver into a theorem prover.
- We demonstrates the effectiveness of the approach by proving global convergence for a state-of-the-art AMS design.
- Benefits we can get from analytical approach:
  - Ranges for initial states, parameters and etc.
  - Proofs are easy to extend. (C rational values, V with some uncertainty)

References

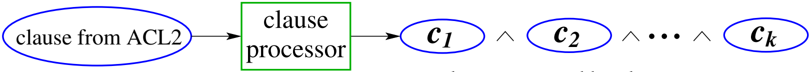
- Ghiath Al-Sammane, Mohamed H Zaki, and Sofiène Tahar, *A symbolic methodology for the verification of analog and mixed signal designs*, Proceedings of the conference on Design, automation and test in Europe, EDA Consortium, 2007, pp. 249–254.
- J. Crossley, E. Naviasky, and E. Alon, *An energy-efficient ring-oscillator digital pll*, Custom Integrated Circuits Conference (CICC), 2010 IEEE, 2010, pp. 1–4.
- Leonardo De Moura and Nikolaj Bjørner, *Z3: an efficient smt solver*, Proceedings of the Theory and practice of software, 14th international conference on Tools and algorithms for the construction and analysis of systems (Berlin, Heidelberg), TACAS'08/ETAPS'08, Springer-Verlag, 2008, pp. 337–340.
- Matt Kaufmann and J. S. Moore, *An industrial strength theorem prover for a logic based on common lisp*, IEEE Trans. Softw. Eng. **23** (1997), no. 4, 203–213.

Technical issues: reals vs. rationals



- Challenge: ACL2 has rationals and Z3 has reals.
  - In ACL2, $\neg\exists x.\ x^2 = 2$ is a theorem.
  - In Z3, $\exists x.\ x^2 = 2$ is a theorem.
- Solution: only use Z3 to prove propositions where all variables are universally quantified.

Technical issues: typed vs. untyped



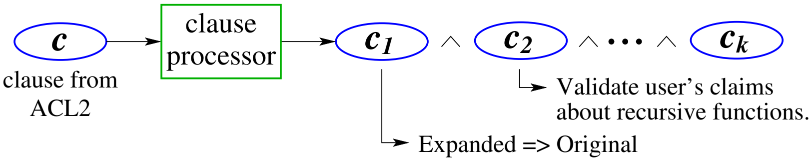

```
(implies (and (rationalp x)
              (rationalp y)
              (rationalp z))
         (c x y z))
```

clauses returned by clause processor:

$$c_1 \wedge c_2 \wedge \cdots \wedge c_k \Rightarrow \mathbf{c(x,y,z)}$$

- Challenge: ACL2 is untyped but Z3 is typed.
- Solution: user adds type assertions to antecedent.
  - These are almost always needed anyways.
  - This requirement is not a significant burden.

Techinical issues: user defined functions



- Challenge:
  - ACL2 supports arbitrary lisp functions.
  - Z3 functions are more like macros (no recursion).
- Solution:
  - Set up translation for a basic set of functions.
  - Expand non-recursive functions.
  - Expand recursive functions to bounded depth.
  - Expansion done on ACL2's representation: can verify correctness.

Other issues:

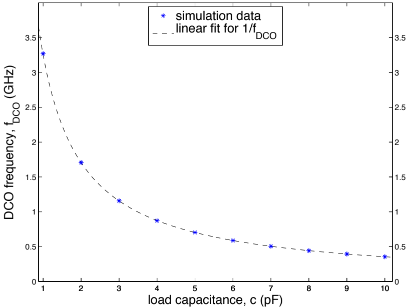
- Claims can contain non-polynomial terms.
  - Replace offensive subexpression with a variable.
  - User adds constraints about the variable.
  - These constraints are returned as clauses for ACL2 to prove.
- ACL2 may need hints to discharge clauses returned from the clause processor.
  - Solution: nested hints.
  - These hints tell the clause processor what hints to attach to returned clauses.
- These features provides a very flexible back-and-forth between induction proofs in ACL2 and handling the details of the algebra with Z3.

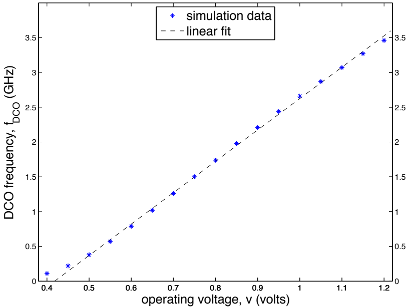
Modeling the digital PLL

$$
\begin{aligned}
c(i+1) &= \min(\max(c(i) + g_c \operatorname{sgn}(\phi), c_{\min}), c_{\max}) \\
v(i+1) &= \min(\max(v(i) + g_v(c_{center} - c(i)), v_{\min}), v_{\max}) \\
\phi(i+1) &= \operatorname{wrap}(\phi(i) + (f_{dco}(c(i), v(i)) - f_{ref}) - g_\phi \phi(i)) \\
f_{dco}(c, v) &= \tfrac{1+\alpha v}{1+\beta c} f_0 \\
\operatorname{wrap}(\phi) &= \operatorname{wrap}(\phi + 1), \quad \text{if } \phi \le -1 \\
&= \phi, \qquad\qquad\quad\ \ \text{if } -1 < \phi < 1 \\
&= \operatorname{wrap}(\phi - 1), \quad \text{if } 1 \le \phi
\end{aligned}
$$

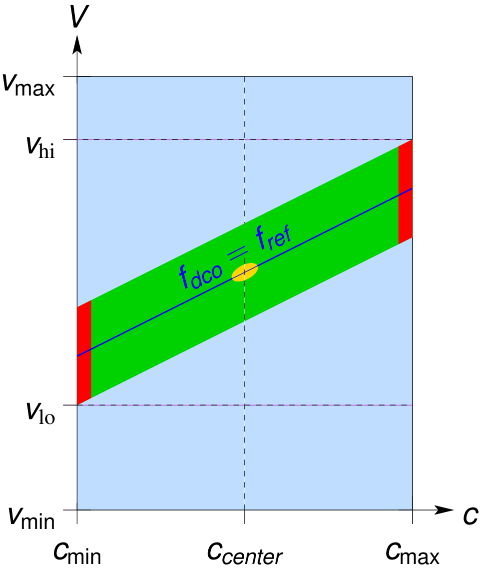
- By simulation we get the model for *fdco*.
- Eliminate differential equation.
- This approach is proposed in [ASZT07].

Simulation results for approximating dco





Basic structure of proof



Future work

- Extend our integration and contribute to the ACL2 community.
  - Automatic function expansion.
  - Automatically generated hints.
  - Checked counterexample reports.
- Automate AMS proofs.
- Example problems from other physical domains: medical, machine learning and etc.