

Experiments in Open Architectures for Formal Verification Tools

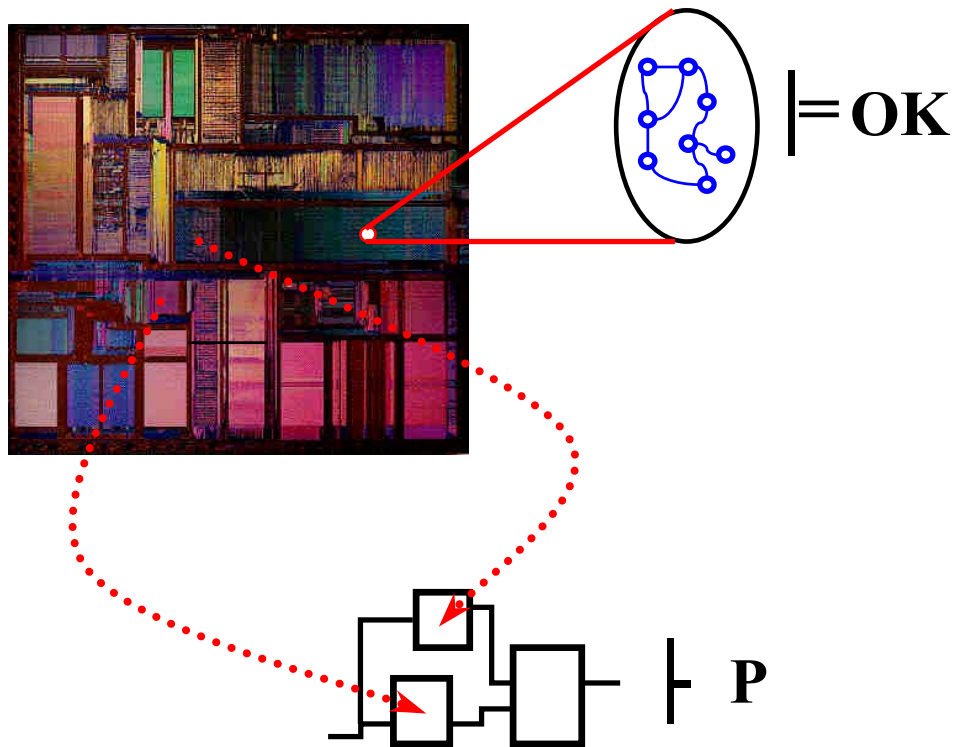
Tom Melham
PROSPER Project Coordinator



UNIVERSITY
of
GLASGOW



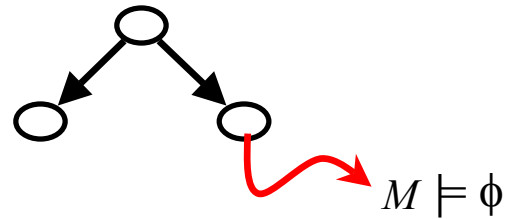
Formal Verification Today



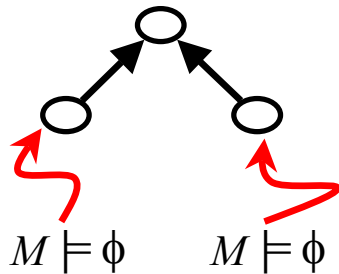
- Model-based:
 - PTL/FSMs
 - automatic, ‘brute force’
 - equivalence checking
 - property checking
 - FSM traversal
- Deductive:
 - higher-order logic
 - expressive, undecidable
 - symbolic deduction
 - abstract data modeling
 - induction



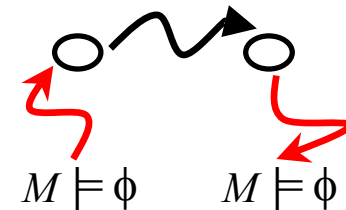
Emerging Combined Approaches



Model Checker as Decision Procedure



Bottom-Up Composition



Cooperating



Formal Verification Tools

- Current situation:
 - not integrated
 - internally monolithic
 - externally user-centric
 - expert user interaction

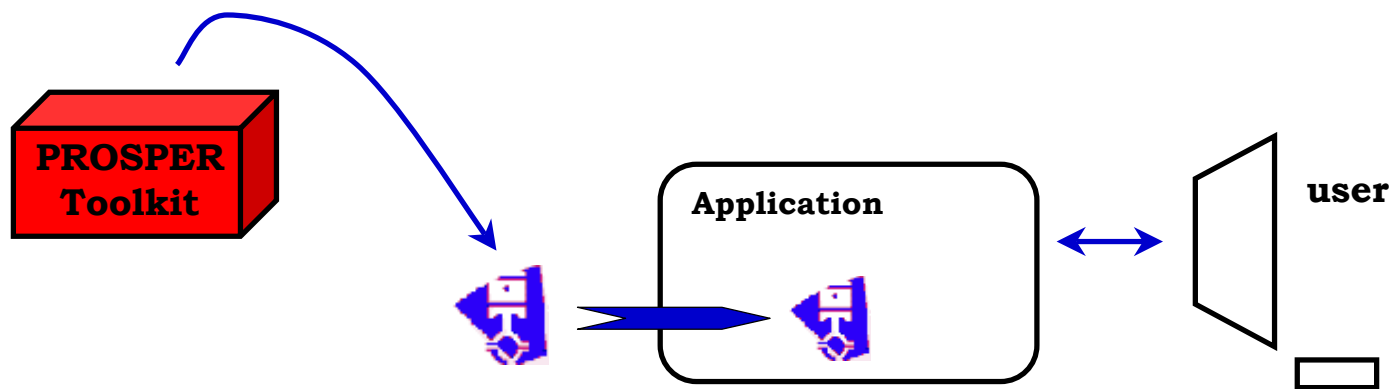
- Vision:
 - integrated proof engine
 - component-based
 - API-driven
 - hidden or push-button
 - user-oriented guidance

Design tools with ‘proof-engine inside’



The PROSPER Project

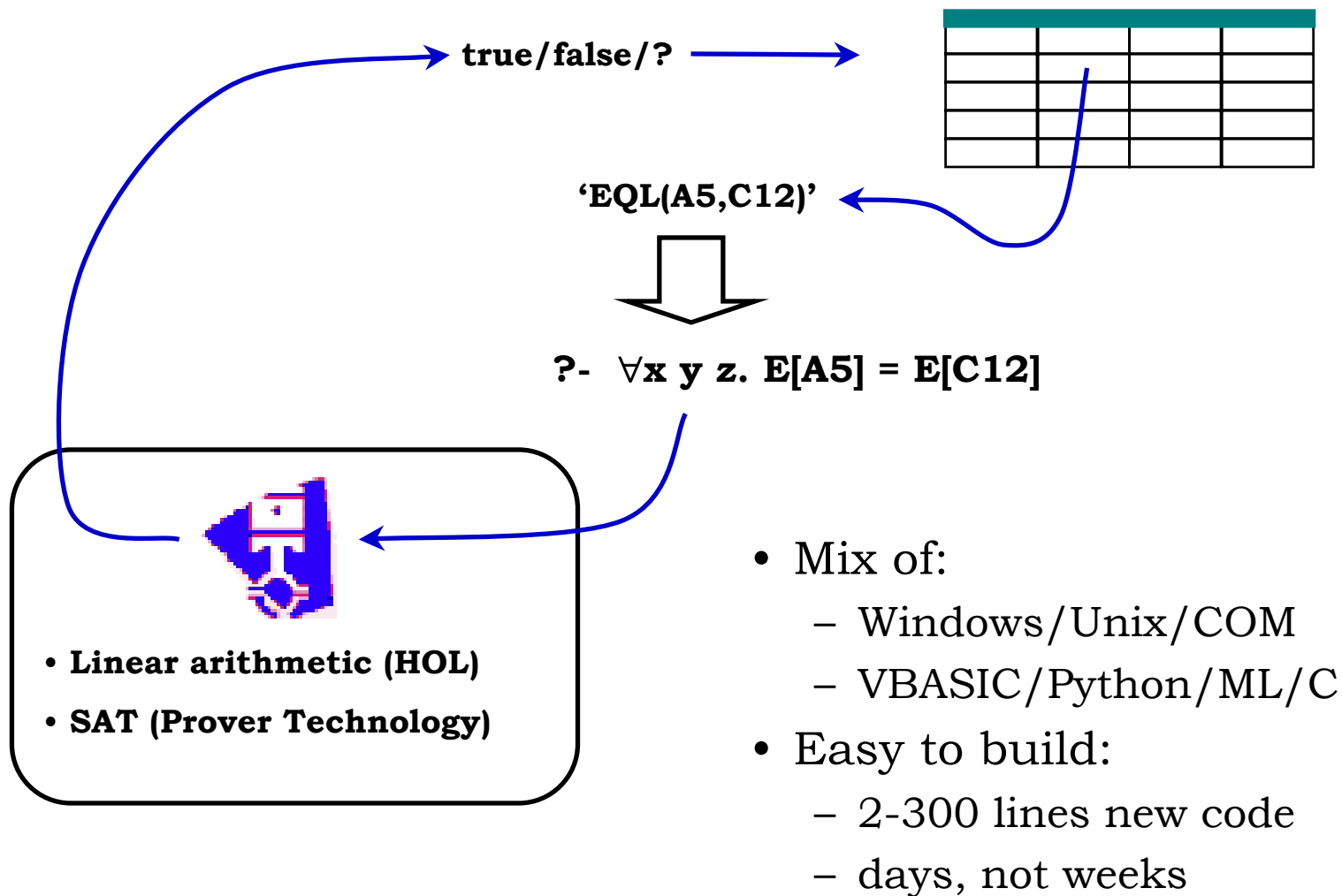
- Open proof-tool architecture



- component toolkit for custom proof-engines
- hardware and software applications
- natural user interfaces/interaction

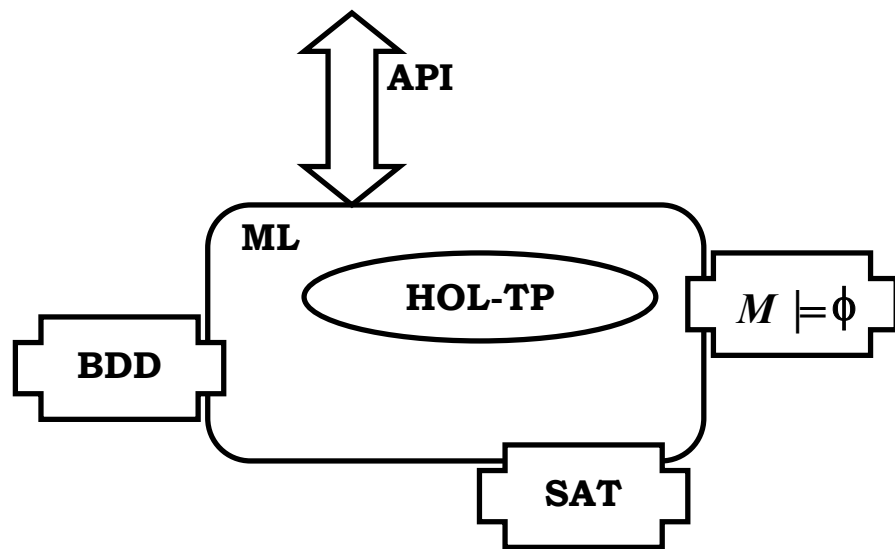


Toy Example - Excel





The PROSPER Architecture

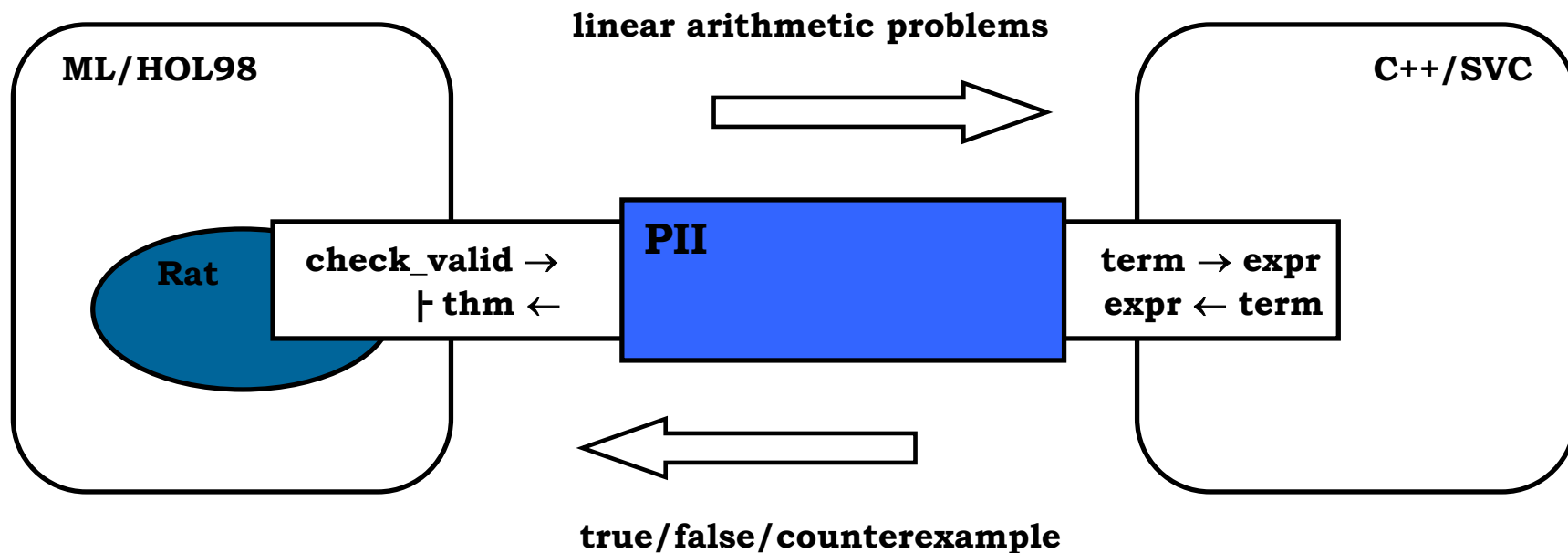


- Plug-in components:
 - BDDs, SAT procedure
 - SVC, SMV, ACL2
 - HOL libraries

- Role of ML and theorem prover:
 - orchestrate verifications (scripting language)
 - semantics for combining results (higher order logic)
 - co-operating deduction and FSM algorithms



A Bit More Detail



- PROSPER Toolkit:
 - data transport
 - control/interrupt handling
 - languages: C++, ML, Java,...

- Research issues:
 - system engineering
 - semantic coherence
 - debugging/counterexamples



Two ‘Speculative’ Examples

• Natural Language:

- NL interface to LTL/CTL
- Graphical feedback
- Driving model checkers
- ‘Lint’ for protocol specs
- SMV Model checker + HOL

Edinburgh and Tübingen:

Lex Holt
Dirk Hoffman
Ewan Klein
Thomas Kropf

• SoC Verification:

- SLI verification platform
- Heterogeneous proof tools
- Whole-system reasoning
- Re-use of proof effort
- ACL2 + HOL + SMV

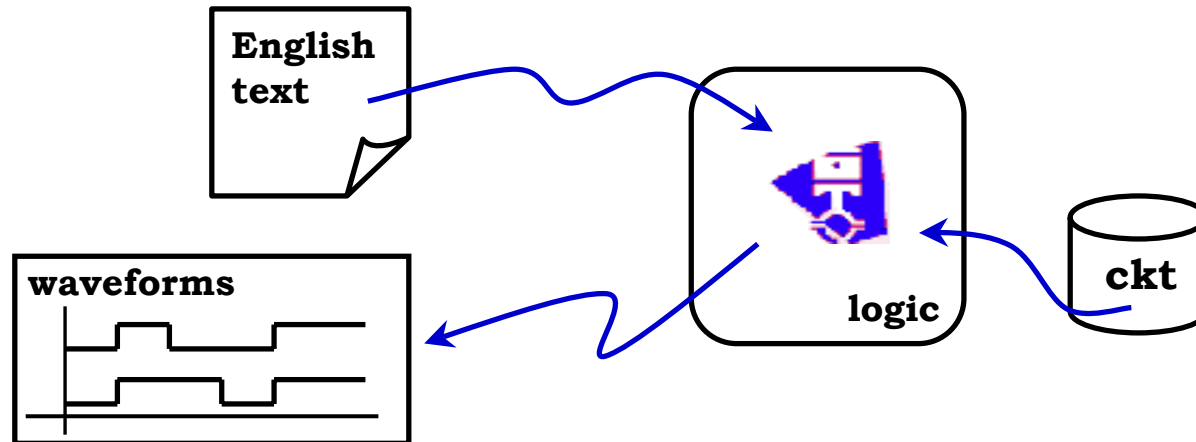
Glasgow and Cambridge:

Kong Woei Susanto
Mark Staples
Tom Melham
Mike Gordon



Natural Language - Motivation

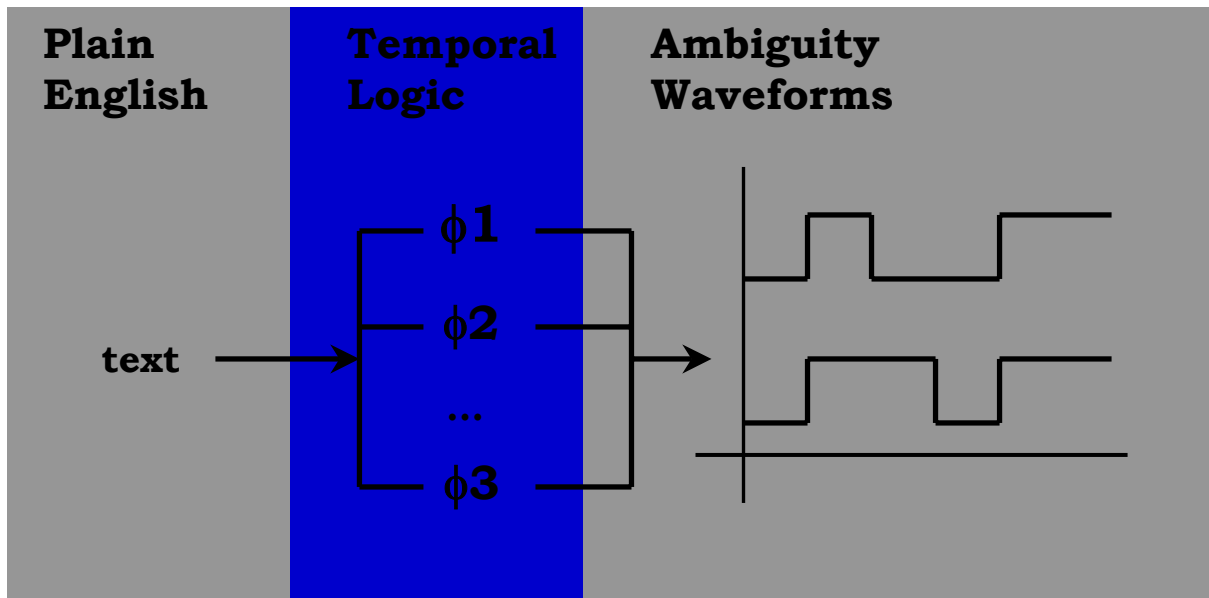
- Integrate temporal logic reasoning into a more natural interaction style:



- drive a model checker for property checking?
- check natural language specs for ambiguity?



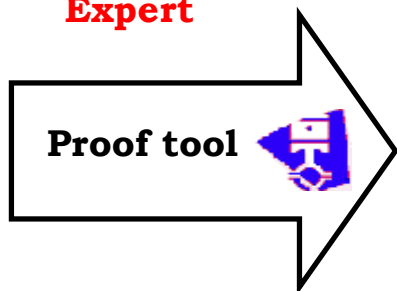
Ambiguity and Proof Automation



Domain Expert

Temporal Logic Expert

Domain Expert

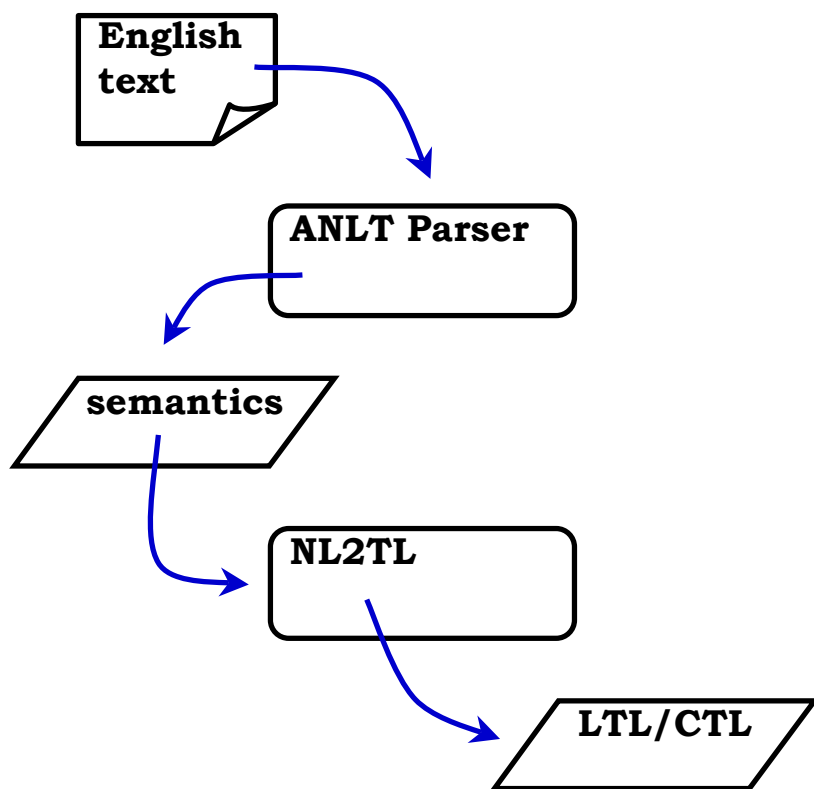


- NL ambiguous, vague
- use automatic proof tool to bridge the gap

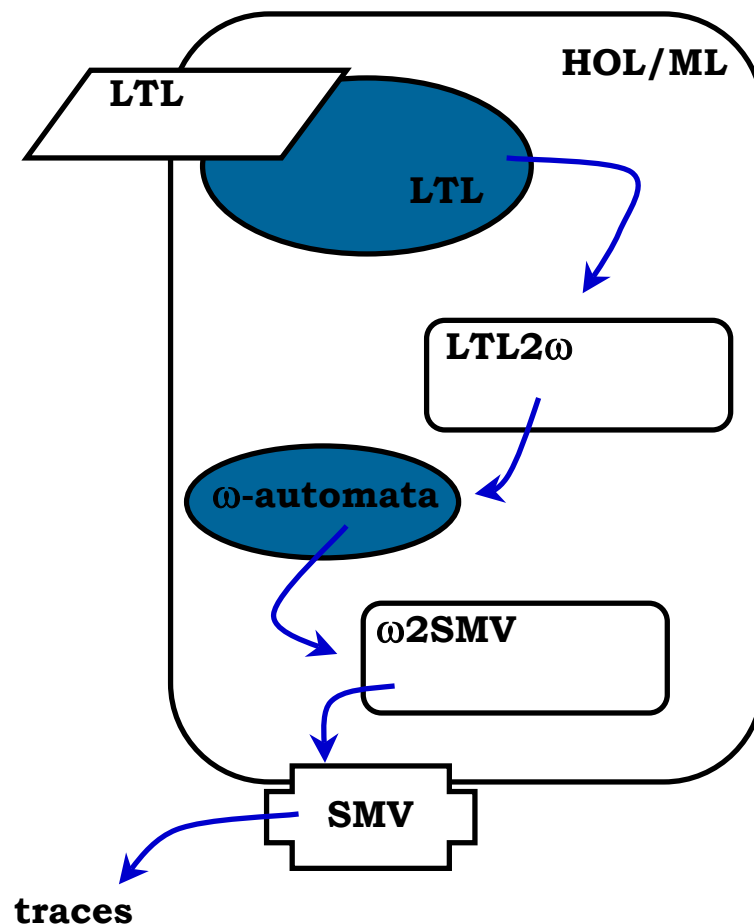


System Architecture/Flow

- NL front-end:



- Proof tool back-end





LTL Formulas

- Temporal logic operators:

$G(\phi)$: ϕ holds forever into the future

$F(\phi)$: ϕ holds at some point in the future

$X(\phi)$: ϕ holds at the next point in time

$\phi \text{ U } \varphi$: ϕ holds until φ becomes true

- Examples:

$\text{sig1} \supset X(\text{sig2} \wedge \neg \text{sig1})$

$G(\text{req} \supset F(\text{ack}))$



Conversion Example

- English:

After sig1 is active, sig2 is active for three cycles.

- Parser output:

```
(DECL
  (some (e1) e1
    (AFTER e1 (some (e2) (PRES e2
      (BE e2 (ACTIVE (name sig1) (degree unknown))))
    (3 (x1) (and (p1 x1) (CYCLE x1))
    (some (e3) (PRES e3
      (and (BE e3 (ACTIVE (name sig2) (degree unknown)))
        (FOR e3 x1))))
    (timespan unknown))))
```



Example Continued

- Converted into intermediate form:
AFTER(HIGH(sig1), FOR(HIGH(sig2), 3), TIMESPAN(unknown))
- Converted into semi-logical form:
If(High(sig1), Next(1, For(3, High(sig2))))
- LTL Translation:
 $G(\text{sig1} \supset X(\text{sig2} \wedge X\text{sig2} \wedge XX\text{sig2}))$



Ambiguity Example

- From the corpus of test data:
 - After sig1 becomes active sig2 should not become active until sig3 becomes active.
- Possible parses:
 - $(\text{sig1} \supset X(\neg \text{sig2})) \cup \text{sig3}$
 - $G(\text{sig1} \supset X(\neg \text{sig2} \cup \text{sig3}))$
- Distinguishing trace:
 - $\{\text{sig1}, \neg \text{sig3}\}, \{\text{sig2}, \text{sig3}\}$

Try an example at: <http://www.ltg.ed.ac.uk/prosper/>



Theorem Proving Back-End

- Higher Order Logic:

- $\neg P, P \wedge Q, P \supset Q, \forall x.P, \exists x.P$

- $\forall P.(P\ 0) \wedge (\forall i. P\ i \supset P(i+1)) \supset \forall n. P\ n$

- Functions are ‘first-class’ values:

- $\lambda x. P(x+1)$

- $\forall = \lambda P. P = \lambda x.True$

- LTL in HOL - the *shallow embedding* technique:

LTL	HOL Translation
ϕ	$\lambda t:\text{Nat}. \phi$
$G(\phi)$	$\lambda t.\forall n. \phi(t+n)$
$F(\phi)$	$\lambda t.\exists n. \phi(t+n)$



LTL to ω -automata [Schneider]

- Shallow embedding of ω -automata:

- states are $q_0, \dots, q_n : \text{Nat} \rightarrow \text{Bool}$

- inputs are $i_0, \dots, i_m : \text{Nat} \rightarrow \text{Bool}$

- automaton:

$$\begin{aligned} & \exists q_0 \dots q_n. \phi_I(q_0(0), \dots, q_n(0)) \wedge \\ & \quad \forall t. \phi_R(q_0(t), \dots, q_n(t), i_0(t), \dots, i_m(t), q_0(t+1), \dots, q_n(t+1)) \wedge \\ & \quad \phi_F(q_0, \dots, q_n) \end{aligned}$$

- acceptance condition:

$$\bigwedge_{k \in \{1..a\}} \forall t_1. \exists t_2. \phi_k(q_0(t_1 + t_2), \dots, q_n(t_1 + t_2))$$



Translation

- Convert to ‘definition normal’ form:

$$\exists d_0 \dots d_n. \left(\bigwedge_{i \in \{1..n\}} (d_i = \varphi_i) \right) \wedge \psi$$

- Example:

$$F(G(a)) \supset G(F(a)) \quad \rightsquigarrow \quad \left(\begin{array}{l} d_1 = G(a) \\ d_2 = F(d_1) \\ d_3 = F(a) \\ d_4 = G(d_3) \end{array} \right) \wedge (d_2 \supset d_4)$$



Final Theorem Proving Phase

- Fixed-point equations for LTL operators:

$$G(y = a \wedge X(y)) \equiv G(y = G(a)) \vee G(y = \text{False})$$

- Rewrites to use in transformation:

$$G(y = G(a)) = \underbrace{G(y = a \wedge X(y))}_{\text{transition}} \wedge \underbrace{G(F(a \supset y))}_{\text{acceptance}}$$

$$\forall t. d = a(t) \wedge d(t+1)$$

$$\forall t_1. \exists t_2. d = a(t_1 + t_2) \wedge d(t_1 + t_2)$$

Details at: <http://goethe.ira.uka.de/~schneider/>



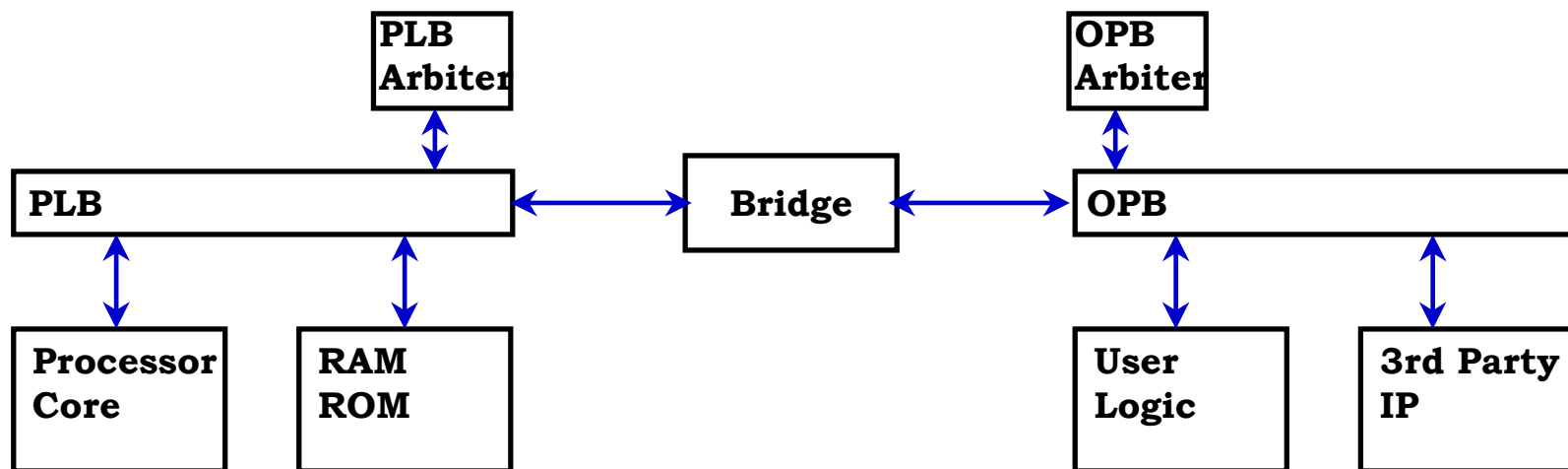
Second Example - FV for SoC?

- SoC/SLI verification:
 - reason with high-level specifications
 - heterogeneous world:
 - hardware and software
 - mixed signal
 - theorem proving almost certainly unavoidable
- pretty much a wide-open research field
- IP blocks:
 - generic blocks
 - parameterized designs.
 - well specified.
 - spec *is* the IP!
 - amortize cost of formal specification and verification over reuse.
- IP block \neq small ASIC!



'Integration Platform' Approach

- Typical SoC integration platform:

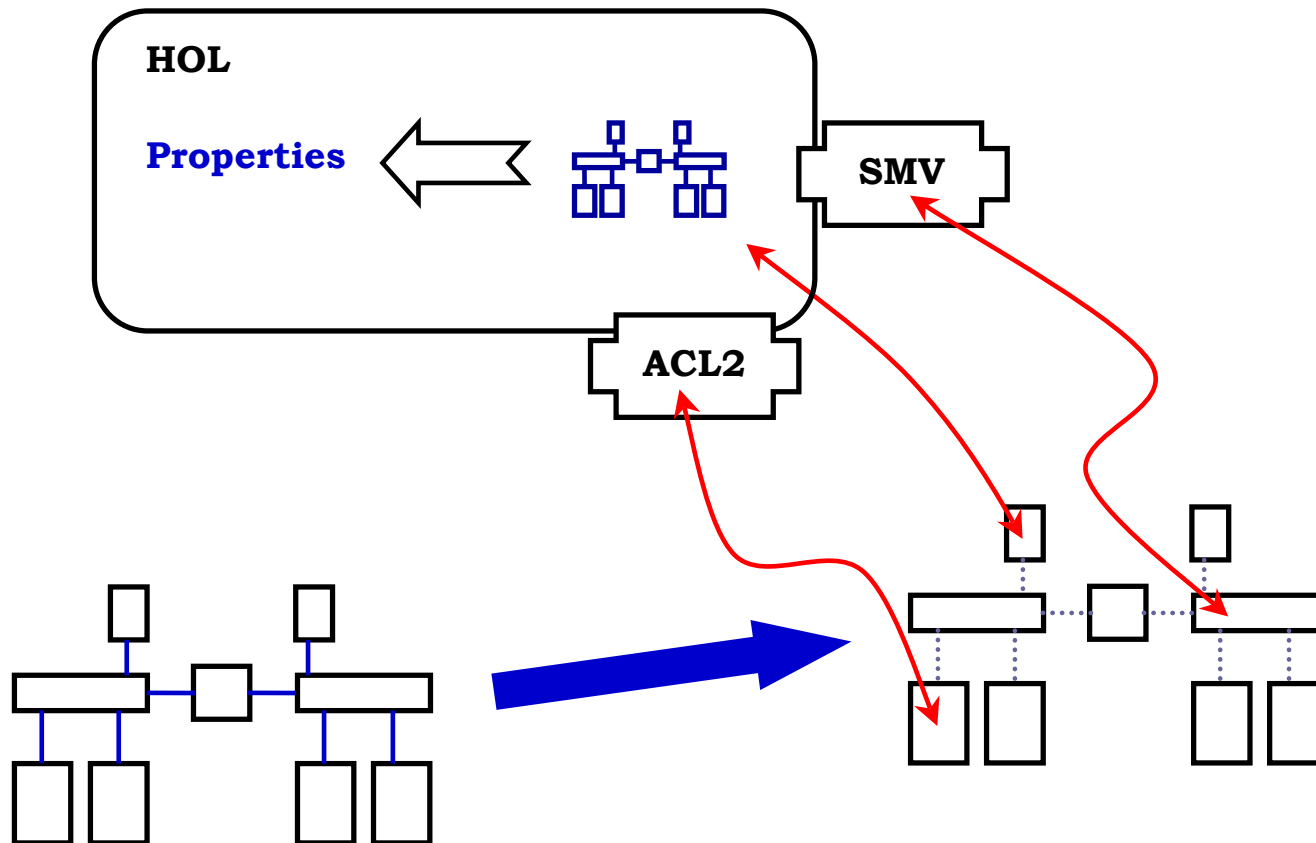


- TI, VSLI Technology, ...
- Cadence, Synopsys, Mentor Graphics, ...
- Validation: HW/SW co-simulation



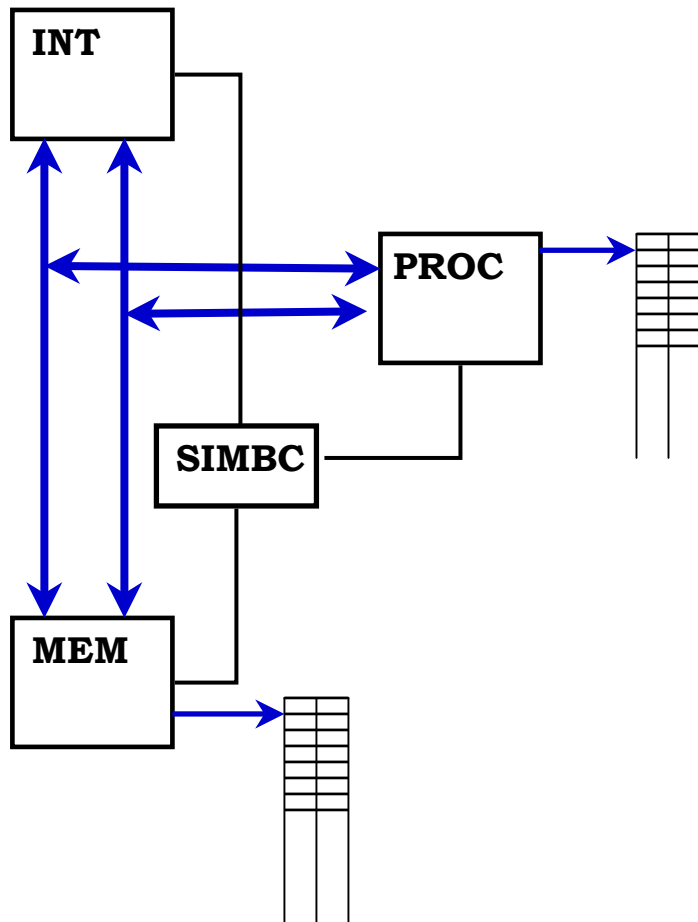
SoC 'Verification Platform'

- Heterogeneous proof environment:





Initial Tiny Example (Susanto)



- Toy ARM in ALC2:
 - quantifier-free 1st order logic
 - fast (formal) simulation
 - later, full ARM7 spec
- SIMBC in SMV:
 - LTL-in-HOL specification
 - Verilog implementation
 - validate with PROSPER flow
- All other models in HOL
- Property derivation:
 - no bus contention
 - ACL2 ‘axiom server’ for HOL
 - top-level proof in HOL



Status and Conclusions

- **NL Tool:**

- ◇ have working prototype
- ◇ results look promising
- ◇ user input now needed

- **SoC Tool:**

- ◇ only just started
- ◇ toy application next
- ◇ then try to scale up
- ◇ definitely 'academic'
study at present

- **General lessons:**

- + vision *is* achievable
- + architecture works
- heavy engineering
research overhead
- suitability of FP/ML
- ? debugging support
- ? embedding support



Acknowledgements

- PROSPER:
 - University of Glasgow
 - University of Cambridge
 - University of Edinburgh
 - University of Tübingen
 - University of Karlsruhe
 - IFAD, Denmark
 - Prover Technology, Sweden
- Funding:
 - EC Esprit Programme
 - 24 person-years, £1.5M
- Excel Demo:
 - Graham Collins (Glasgow)
 - Louise Dennis (Glasgow)
- NL Tool:
 - Lex Holt (Edinburgh)
 - Dirk Hoffman (Tübingen)
 - Klaus Schneider (Karlsruhe)
- SoC Verification Platform
 - Kong Woei Susanto (Glasgow)
- Further information:
<http://www.dcs.gla.ac.uk/prosper>