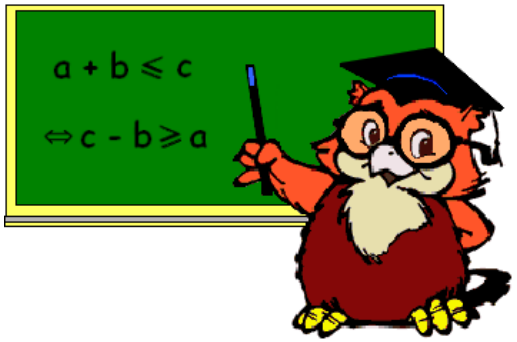
Outils de démonstration automatique et preuve de circuits électroniques

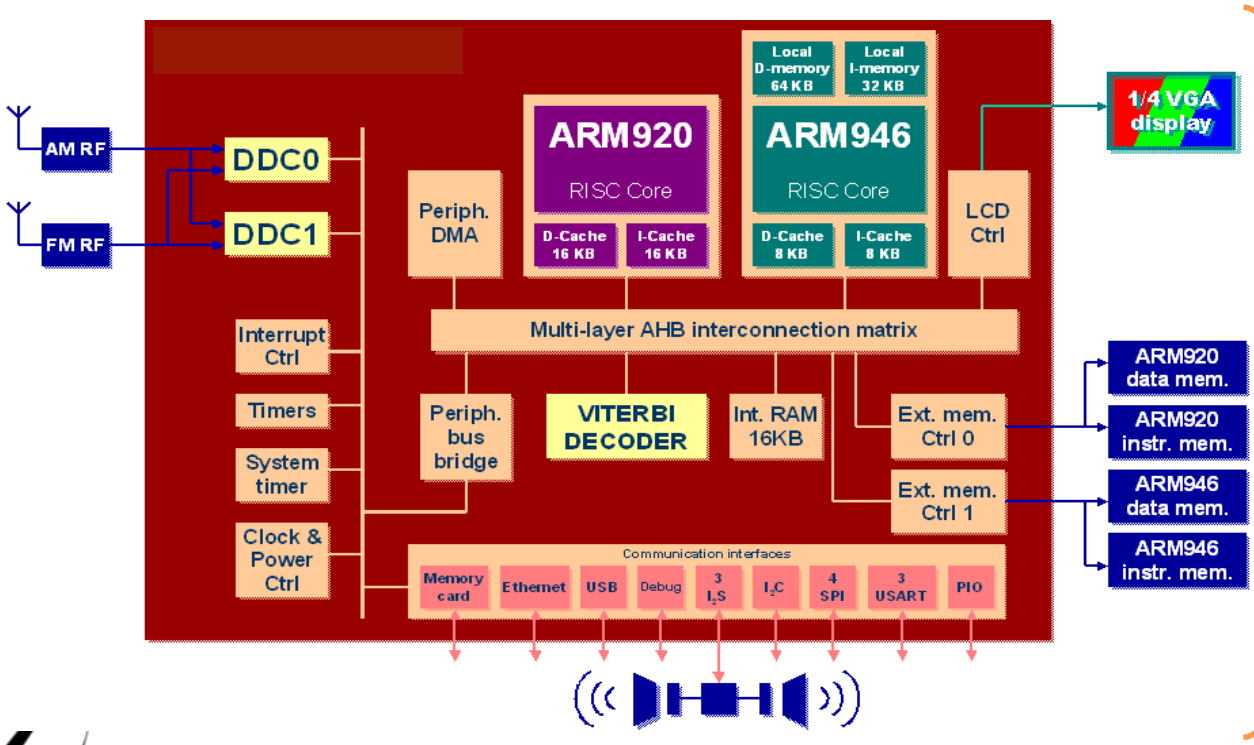


Laurence Pierre

Laboratoire TIMA, Grenoble

PREAMBLE

- Design/validation of embedded applications:

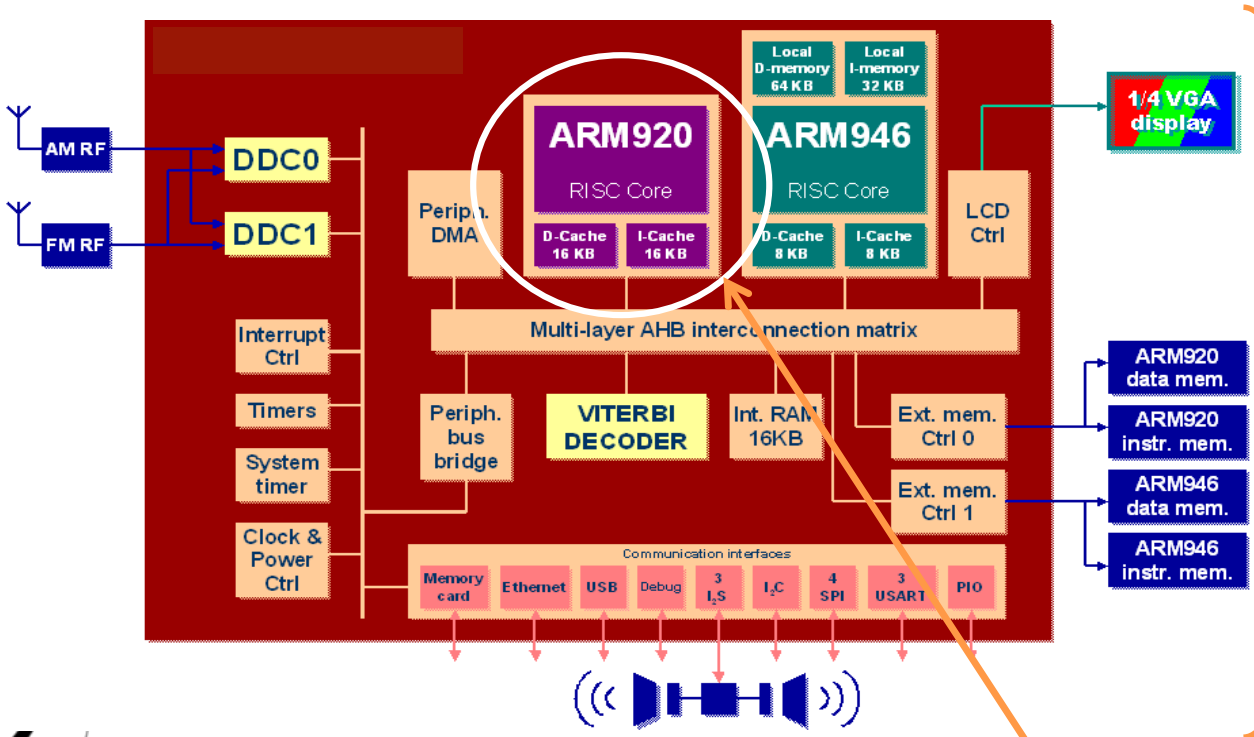


Design/validation for the **system**

Source du dessin : http://www.etis.ensea.fr/~verdier/cours/dimitri_upc_ensea_231106_version_electronique.pdf

PREAMBLE

- Design/validation of embedded applications:



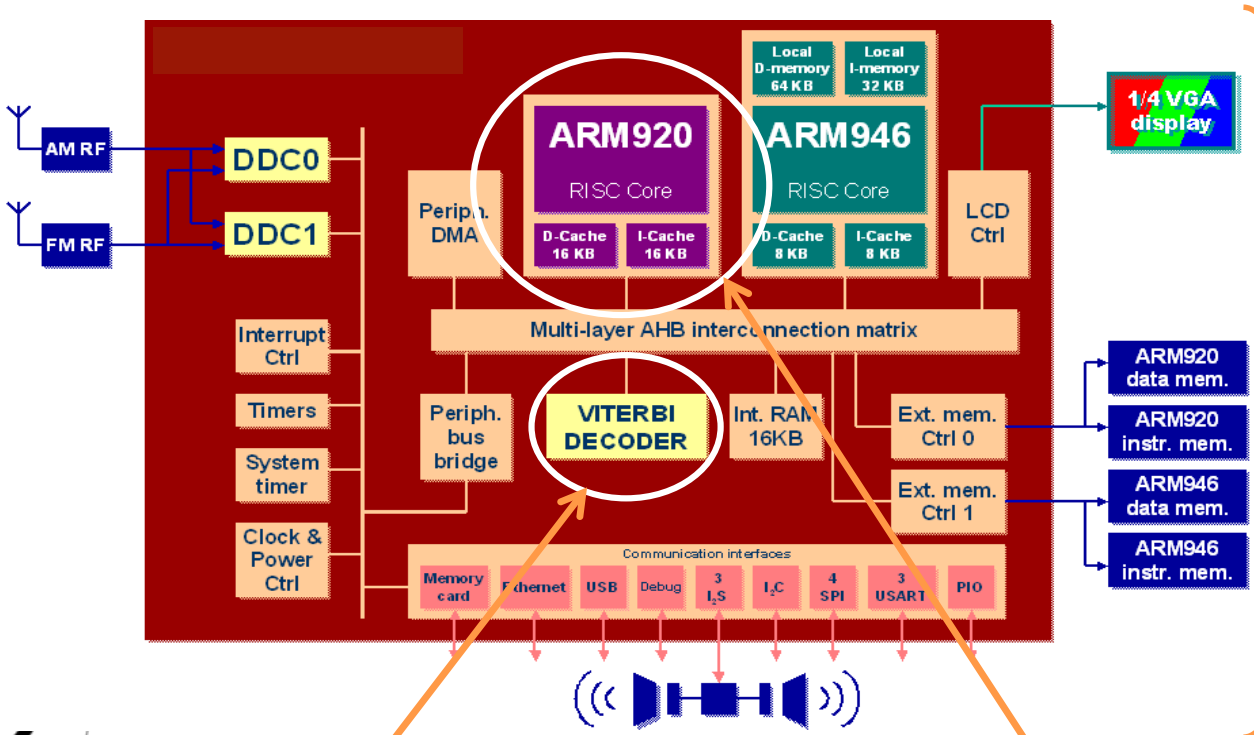
Design/validation for the **system**

Design/validation for the **software**

Source du dessin : http://www.etis.ensea.fr/~verdier/cours/dimitri_upc_ensea_231106_version_electronique.pdf

PREAMBLE

- Design/validation of embedded applications:



Design/validation for the **system**

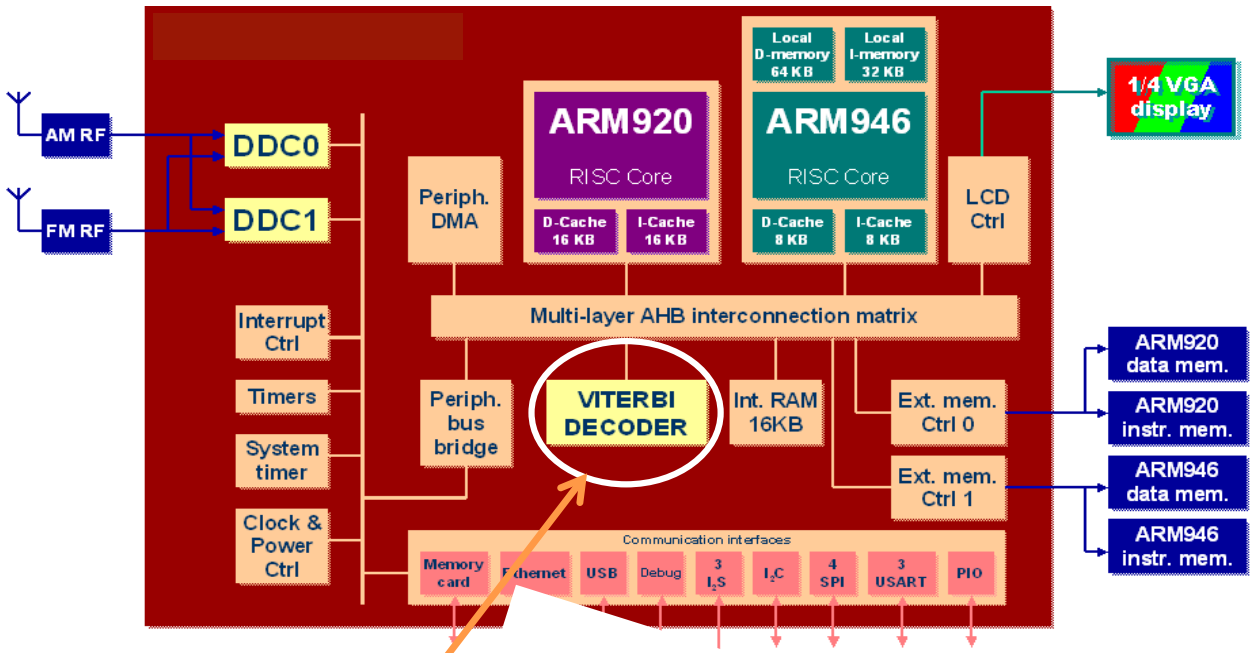
Design/validation for the **hardware components (ex. coprocessor)**

Design/validation for the **software**

Source du dessin : http://www.etis.ensea.fr/~verdier/cours/dimitri_upc_ensea_231106_version_electronique.pdf

PREAMBLE

- Design/validation of embedded applications:



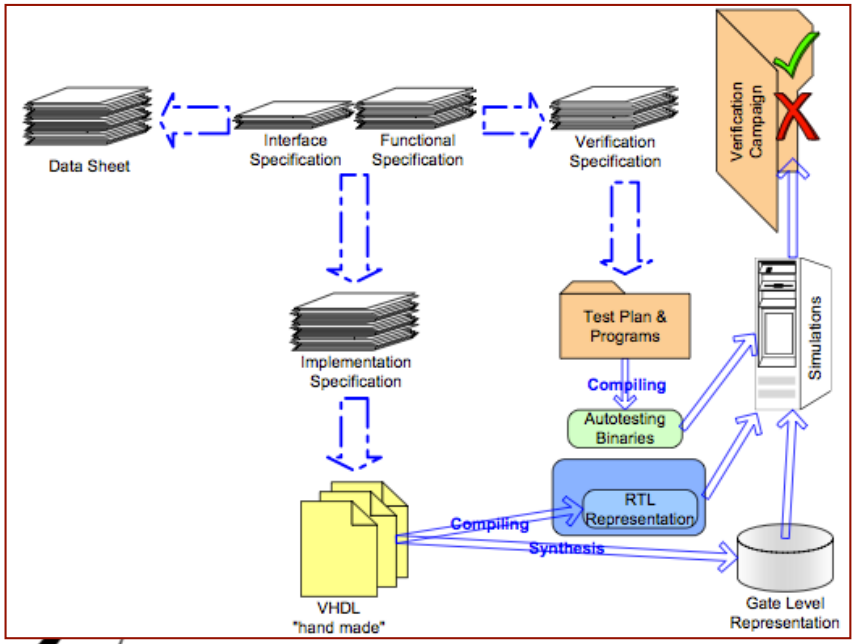
IN THIS PRESENTATION

Design/validation for the hardware components (ex. coprocessor)

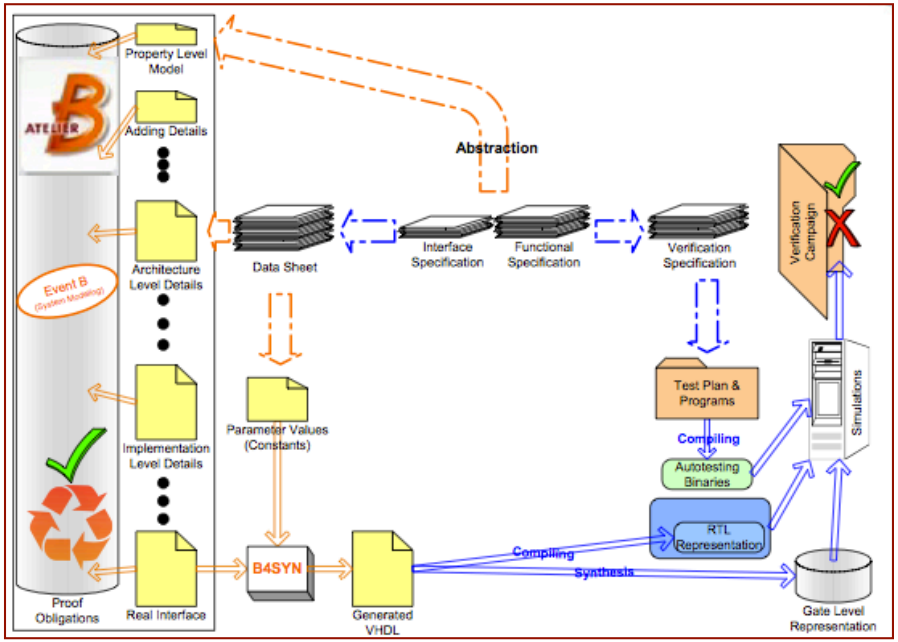
Source du dessin : http://www.etis.ensea.fr/~verdier/cours/dimitri_upc_ensea_231106_version_electronique.pdf

Preamble

- In hardware design, like in software design, two main approaches:



Implementation + a posteriori verification

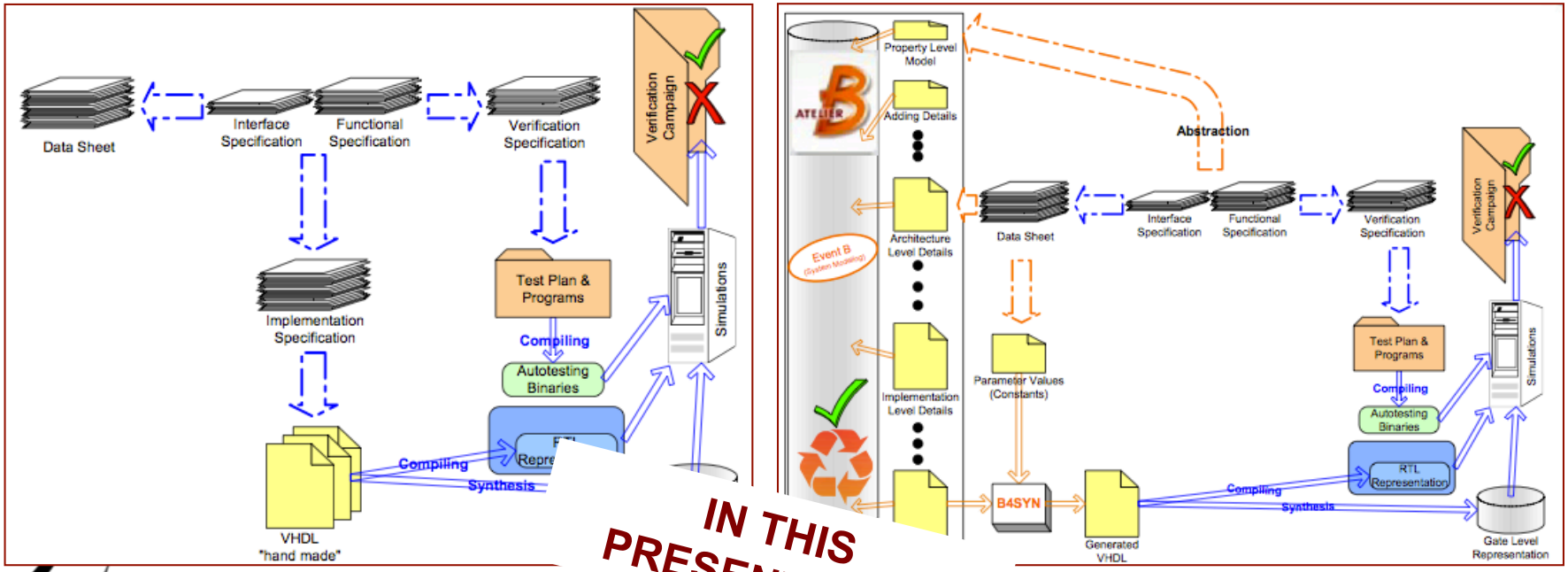


"Correct by construction" implementation

Source des dessins : http://www.methode-b.com/recherche-developpement/forcoment/

PREAMBLE

- In hardware design, like in software design, two main approaches:



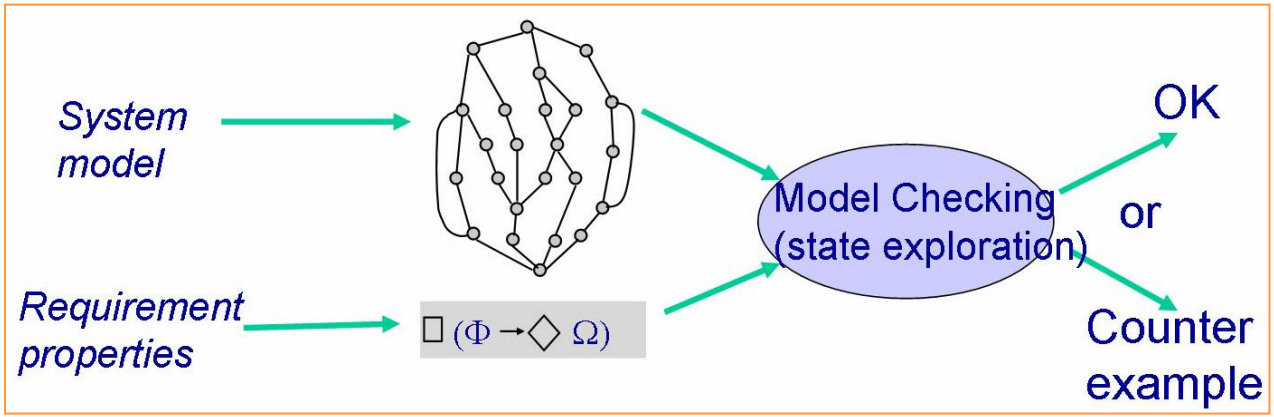
IN THIS PRESENTATION

Implementation + a posteriori verification

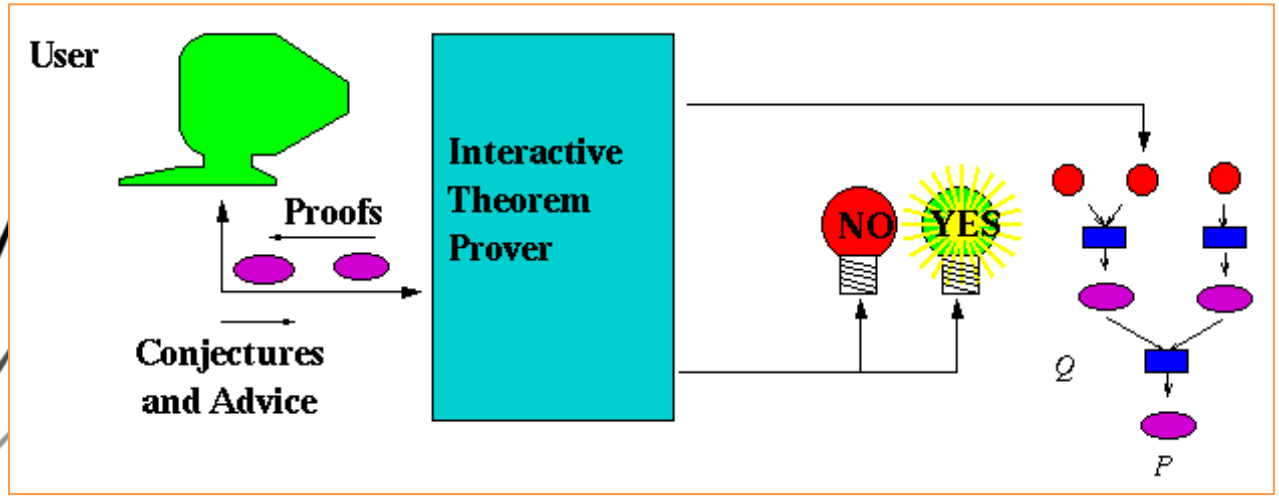
Source des dessins : http://www.methode-b.com/recherche-developpement/forcoment/

PREAMBLE

- And two main approaches for verification:



Algorithmic (model checking)

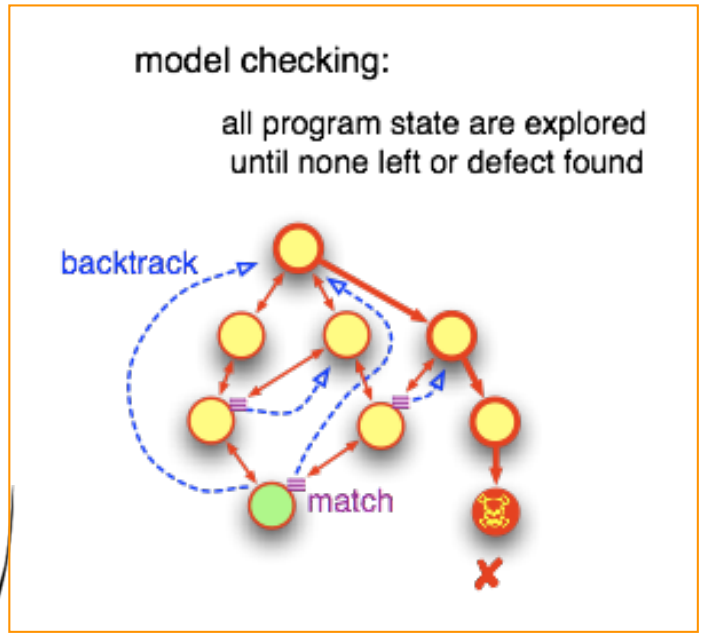


Deductive (theorem provers)

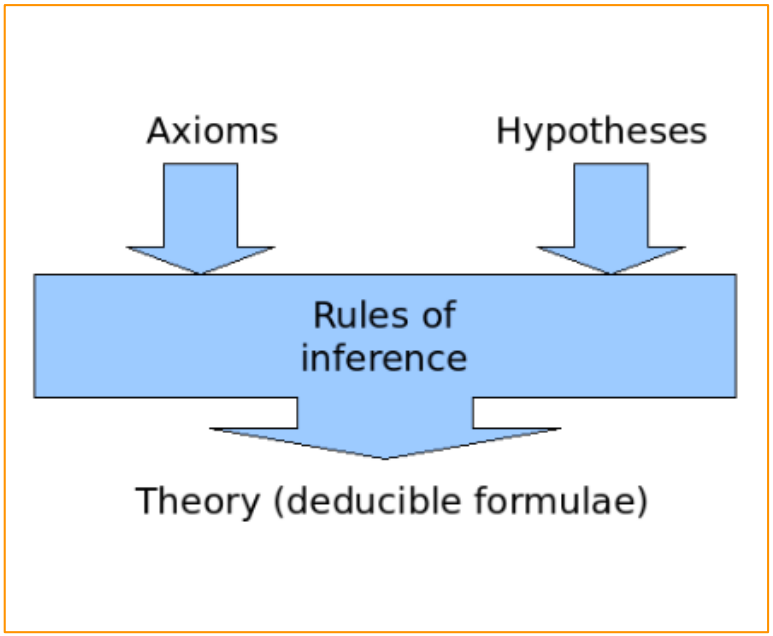
Sources des dessins : http://cs.kaist.ac.kr/~moonzoo/cs750b/index.html, et tutorial ACL2

Preamble

- And two main approaches for verification:



Algorithmic (model checking)

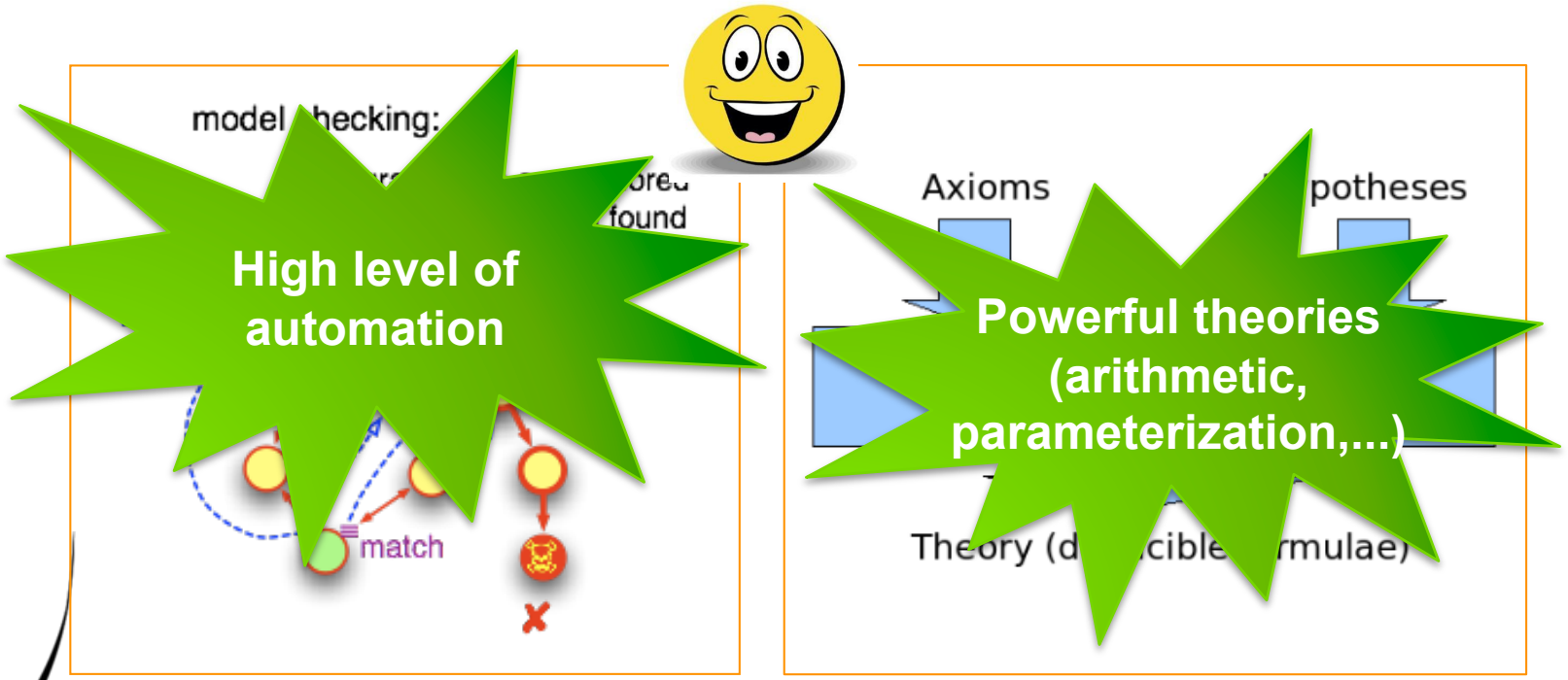


Deductive (theorem provers)

Sources des dessins : http://babelfish.arc.nasa.gov/trac/jpf/wiki/intro/testing_vs_model_checking, http://en.wikipedia.org/wiki/Hilbert_system

Preamble

- And two main approaches for verification:



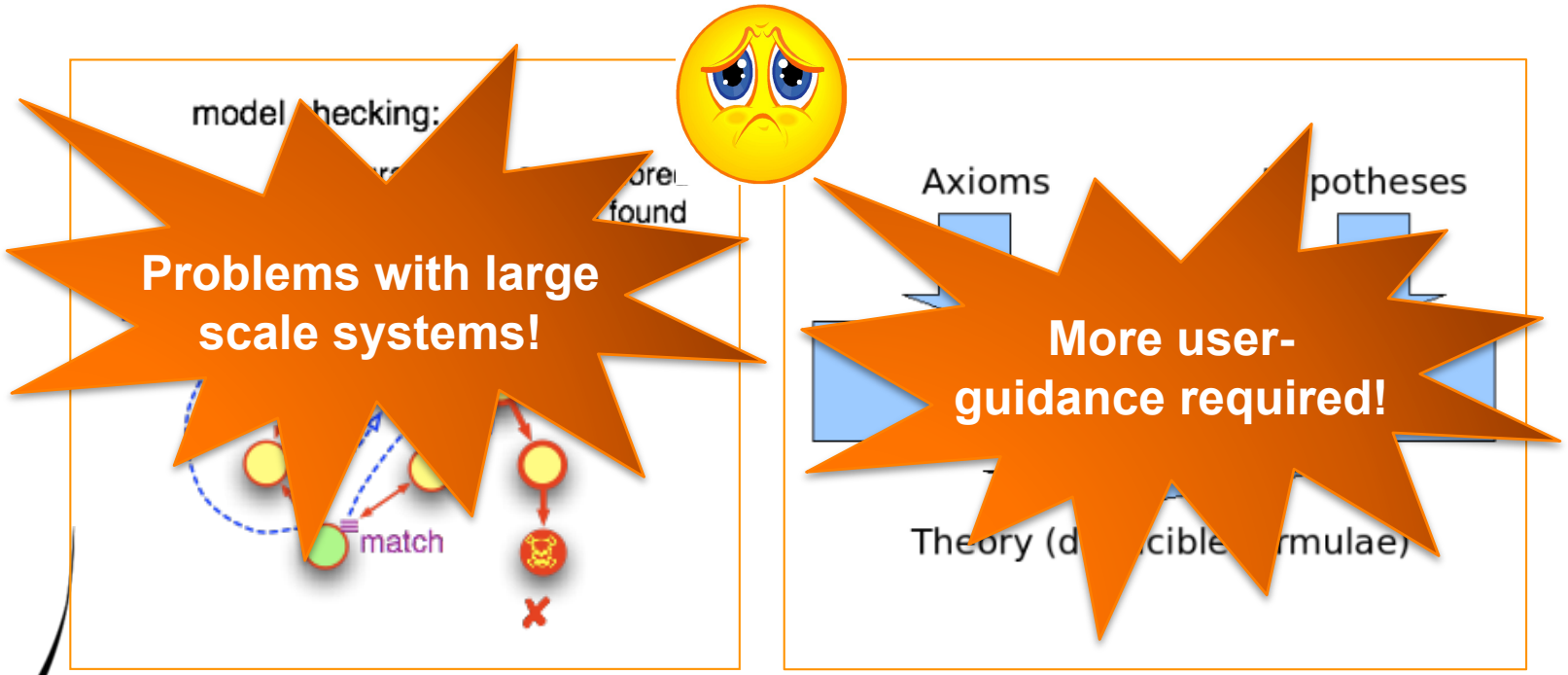
Algorithmic (model checking)

Deductive (theorem provers)

Sources des dessins : http://babelfish.arc.nasa.gov/trac/jpf/wiki/intro/testing_vs_model_checking, http://en.wikipedia.org/wiki/Hilbert_system

Preamble

- And two main approaches for verification:



Algorithmic (model checking)

Deductive (theorem provers)

Sources des dessins : http://babelfish.arc.nasa.gov/trac/jpf/wiki/intro/testing_vs_model_checking, http://en.wikipedia.org/wiki/Hilbert_system

Preamble

- And two main approaches for verification:

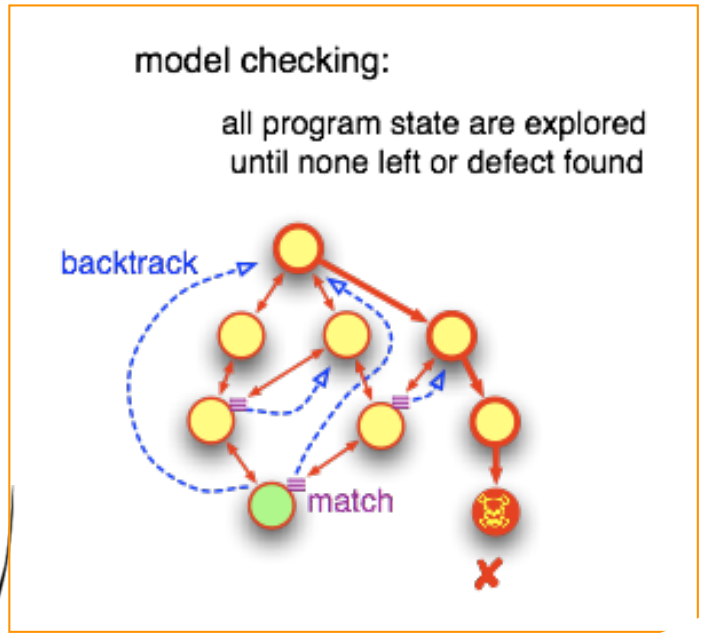
Algorithmic (model checkin

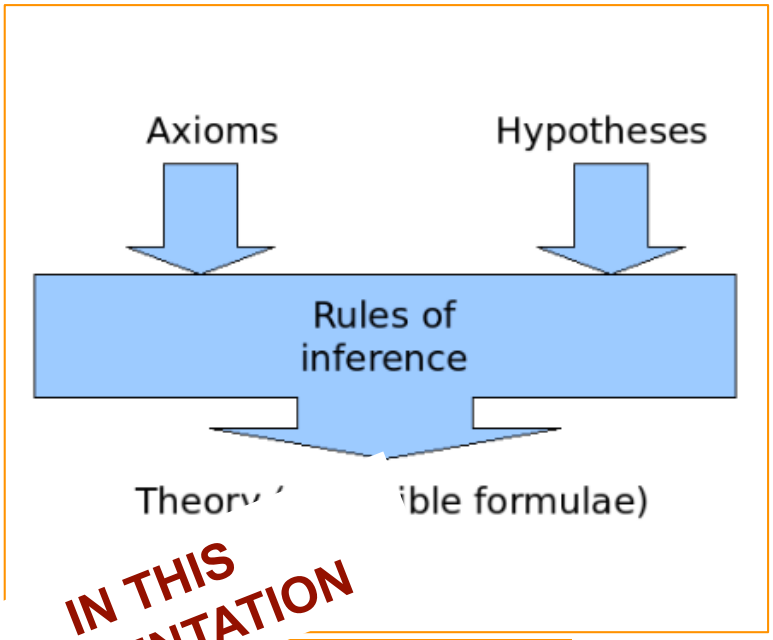
Deductive theorem provers)

Sources des dessins : http://babelfish.arc.nasa.gov/trac/jpf/wiki/intro/testing_vs_model_checking, http://en.wikipedia.org/wiki/Hilbert_system

PREAMBLE

- And two main approaches for verification:





IN THIS PRESENTATION

Deductive (theorem provers)

Sources des dessins : http://babelfish.arc.nasa.gov/trac/jpf/wiki/intro/testing_vs_model_checking, http://en.wikipedia.org/wiki/Hilbert_system

Tools for interactive theorem proving

- Various logics, and various levels of automation:
  - First order logic without quantifiers, with induction : ACL2 *http://www.cs.utexas.edu/~moore/acl2/*

  - Higher order logic :
    - HOL *http://www.cl.cam.ac.uk/research/hvg/HOL/*

    - PVS *http://pvs.csl.sri.com/*

  - Calculus of inductive constructions : Coq *http://coq.inria.fr/*

  - General-purpose : Isabelle *http://www.cl.cam.ac.uk/research/hvg/Isabelle/*

Tools for interactive theorem proving

- First order vs higher order logic
  - In general, in the hardware verification framework higher order logic is less crucial than for other applications
  - It can be used for example to model functions that take *functions over time* as parameters
    - Example: Unit delay element

      D: (time -> bool) x (time -> bool) -> bool

      D(in, out) = ∀ t . out(t+1) = in(t)
  - See for instance "Higher Order Logic and Hardware Verification", T.Melham, Cambridge University Press, 1993

HARDWARE VERIFICATION - A BIT OF HISTORY

- One of the precursors was Mike Gordon:
  - First proofs of digital circuits with the system LCF
  - Then, well-known results with HOL
    - "Why higher-order logic is a good formalism for specifying and verifying hardware", M.Gordon, Formal Aspects of VLSI Design, 1986
- One of the famoust proofs with HOL

  - The VIPER microprocessor: designed for safety critical applications
  - First commercial microprocessor whose design is claimed "proven correct" → controversy
    - "The notion of proof in hardware verification", A.Cohn, Journal of Automated Reasoning, 1989

HARDWARE VERIFICATION - A BIT OF HISTORY

- PVS:
  - For hardware verification: bit-vector library, linear arithmetic, simplification based on BDD,...
  - Proof of a pipelined microprocessor
    - "Effective theorem proving for hardware verification", D.Cyrluk, S.Rajan, N.Shankar, M.Srivas, Proc. TPCD'94
- Coq:
  - Efforts towards the use of Coq for circuits:
    - "Circuits as streams in Coq: verification of a sequential multiplier", C.Paulin-Mohring, Research report, LIP, 1995
    - "Certifying circuits in Type Theory", S.Coupet-Grimal, L.Jakubiec, Formal Aspects of Computing, 2004

ACL2 AND HARDWARE VERIFICATION

- Brief overview of ACL2:
  - A Computational Logic for Applicative Common Lisp
  - First order logic without quantifiers, with equality
  - Total, recursive functions - Induction principle
  - Elementary example

```
(defun times (i j)
   (if (and (integerp i) (<= 0 i))
       (if (equal i 0)
           0
           (+ j (times (1- i) j)))
       0))
(defthm times-add1
   (implies (and (integerp x) (<= 0 x)
                 (integerp y) (<= 0 y))
            (equal (times x (1+ y))
                   (+ x (times x y)))))
```

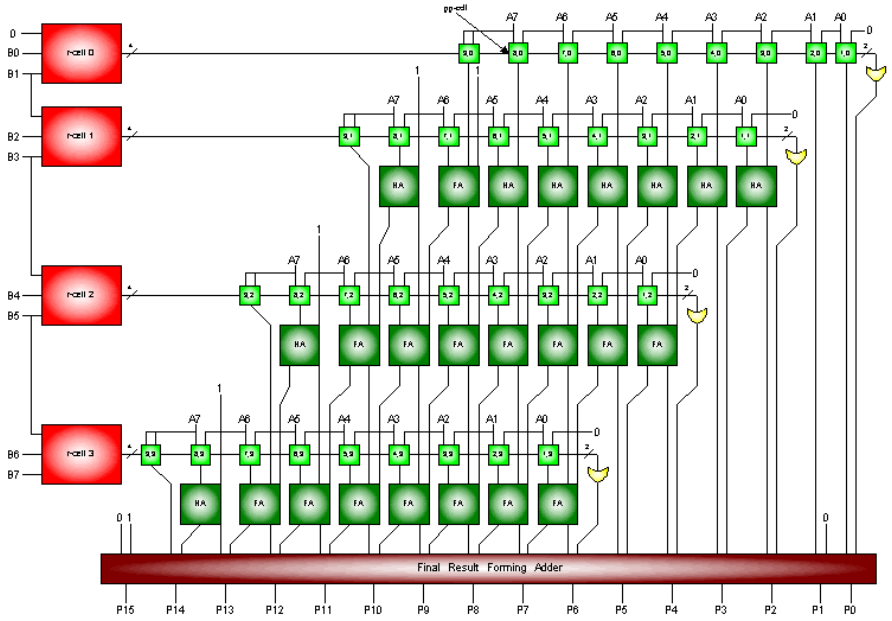
induction on x

ACL2 AND HARDWARE VERIFICATION

- Advantage of its powerful induction mechanism - inductive proofs on recursive functions that can model:
  - Parameterized regular (arithmetic) structures
    - Examples: adders, multipliers, dividers,...
    - Parameterized proofs: N-bit components
    - Verification against a specification at the arithmetic level, use of conversion functions
  - Sequential circuits
    - Recursion models how the state is updated every clock cycle
    - Finite state machine (behavioural view) or structural view

ACL2 AND HARDWARE VERIFICATION

- Induction for parameterized structures:
  - Example : parallel multiplier



Rough idea:

```
A is a bit-vector
and B is a bit-vector
and size(A) = size(B)
and A is not empty
⇒
bv-to-nat(Mult(A, B))
      = times(bv-to-nat(A), bv-to-nat(B))
```

"VHDL Description and Formal Verification of Systolic Multipliers", L.Pierre, Proc. CHDL'93

ACL2 AND HARDWARE VERIFICATION

- Among the seminal works (with the Boyer-Moore theorem prover): "FM8501: a verified microprocessor", Warren Hunt, PhD thesis, 1986
  - Verification of a microprocessor designed by W.Hunt, simple but realistic
    - Definition of the "FM8501 hardware interpreter": function `big-machine` that models sequential logic (parameters are registers and memory, updated on each recursive call)
    - Definition of the FM8501 formal specification: function `soft` at the instruction level (an instruction is interpreted on each recursive call)
    - Verification: any state "computed" by `soft` can be "computed" by `big-machine`

ACL2 AND HARDWARE VERIFICATION

- Details and improvements in: "Symbolic simulation: an ACL2 approach", J Moore, Proc. FMCAD'98
  - Emphasizes the fact that ACL2 provides both a theorem prover and an execution engine
  - Illustrated with a simple example ("small machine")
    - Model for the state (registers and memory) of this machine
    - Each instruction modeled by a function that describes how the state is transformed by its execution
    - Instruction interpreter

ACL2 AND HARDWARE VERIFICATION

- Details and improvements in: "Symbolic simulation: an ACL2 approach", J Moore, Proc. FMCAD'98
  - Example: product of i and j (in mem[0] and mem[1])

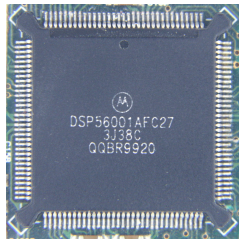
```
(defun times-program nil
  '(times (movi 2 0)      ;   mem[2] <- 0
          (jumpz 0 5)     ;   if mem[0]=0, go to 5
          (add 2 1)       ;   mem[2] <- mem[1] + mem[2]
          (subi 0 1)      ;   mem[0] <- mem[0] - 1
          (jump 1)        ;   go to 1
          (ret)))         ;   return to caller
```

  - This "program" can be executed with actual values
  - And it is possible to prove that

    *For all state s_0 such that mem[0]=i and mem[1]=j, after 4i+4 steps, mem[2] = i*j*

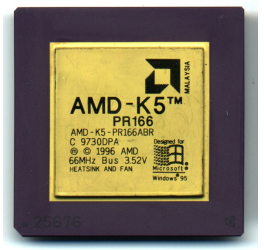
ACL2 AND HARDWARE VERIFICATION

- Other results in "ACL2 theorems about commercial microprocessors", B.Brock et al, Proc. FMCAD'96
  - Validation of a Motorola DSP (Digital Signal Processor)
  - 6 memories, 252 registers, pipelined processor
  - Similar modeling
  - Significant part of the proof: equivalence between the pipelined implementation and a non-pipelined abstraction
  - 31 man-months effort



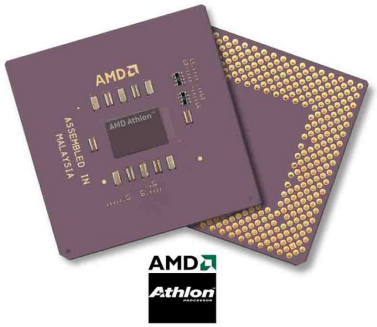
ACL2 AND HARDWARE VERIFICATION

- Verification for AMD microprocessors
  - "A Mechanically Checked Proof of the Correctness of the Kernel of the AMD5K86 Floating-Point Division Algorithm", J Moore, T.Lynch, M.Kaufmann, IEEE Trans. on Computers, 47(9), Sept. 1998
    - Libraries for floating point arithmetic (rational numbers) - "Formal verification of floating-point RTL at AMD using the ACL2 theorem prover", D.Russinoff, M.Kaufmann, E.Smith, R.Sumners, Proc. IMACS'2005
    - Verification that the microcode of the AMD K5 microprocessor correctly implements the IEEE floating point division



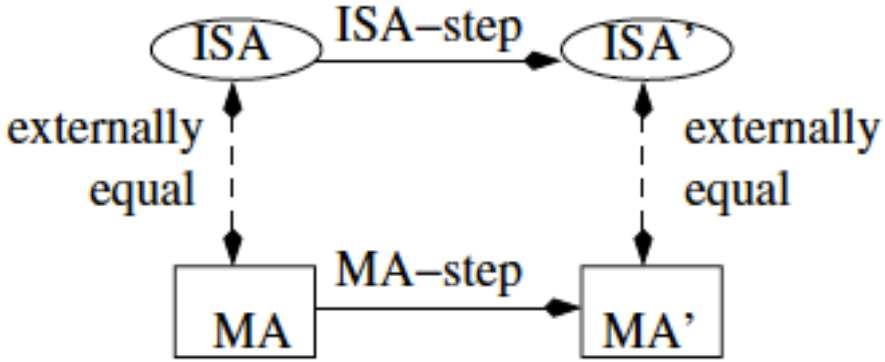
ACL2 AND HARDWARE VERIFICATION

- Verification for AMD microprocessors
  - "A Mechanically Checked Proof of IEEE Compliance of a Register-Transfer-Level Specification of the AMD K7 Floating Point Multiplication, Division and Square Root Instructions", D.Russinoff, LMS Journal of Computation and Mathematics, 1998
    - Verification of the floating point multiplication, division, and square root of the AMD K7 microprocessor
    - Equivalence at different abstraction levels: correspondence between bit-vector implementations and arithmetic specifications (IEEE standard 754)



ACL2 AND HARDWARE VERIFICATION

- Proposal for the verification of pipelined machines
  - "Deductive verification of pipelined machines using first-order quantification", S.Ray, W.Hunt, Proc. CAV'2004
    - Microarchitectural implementation / instruction set architecture



    - Pipelined architecture: Skolem witness to relate pipeline states with ISA states

HARDWARE VERIFICATION - OTHER RESULTS

- Verification of Networks on Chip (NoC's)
  - "Deadlock prevention in the AEthereal protocol", B.Gebremichael et al, Proc. CHARME'2005
  - Use of PVS to prove absence of deadlock for (an abstract model of) the network

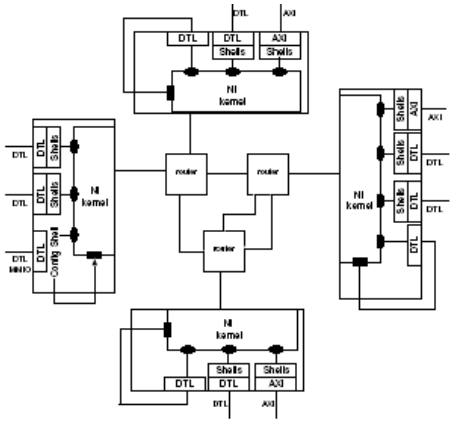


Figure 1. network example

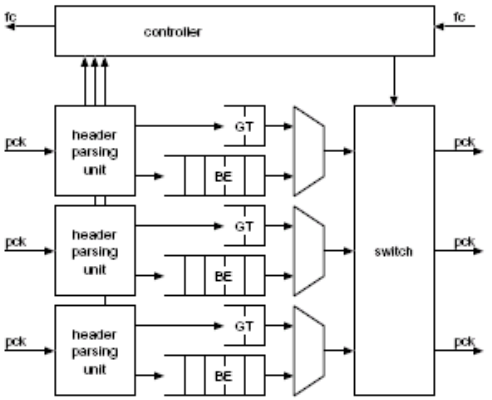
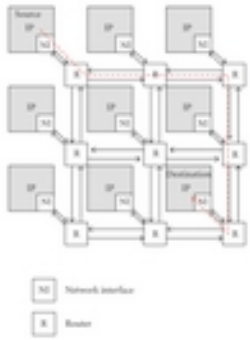


Figure 2. Router architecture

Hardware verification - Other results

- Verification of Networks on Chip (NoC's)
  - "A functional formalization of on chip communications", J.Schmaltz, D. Borrione, Formal Aspects of Computing, 2008

    "A Formal Approach to the Verification of Networks on Chip", D.Borrione, A.Helmy, L.Pierre, J.Schmaltz, EURASIP Journal on Embedded Systems, 2009
    - Meta-model for Networks on Chip - implementation in ACL2
    - Applications: Spidergon (STMicroelectronics), HERMES (PUCRS, Brazil), Nostrum (Royal Institute of Technology, Stockholm),...



Hardware verification - Other results

- Industrial environments: Intel
  - "Relational STE and theorem proving for formal verification of industrial circuit designs", J.O'Leary, R.Kaivola, T.Melham, Proc. FMCAD'2013

    "Replacing testing with formal verification in Intel Core™ i7 processor execution engine validation", R.Kaivola et al, Proc. CAV'2009
    - Use of Symbolic Trajectory Evaluation in the Forte formal verification environment of Intel Corporation
    - Now: Relational Symbolic Trajectory Evaluation (purely logical constraints) with the Goaled theorem prover
    - Example: framework for integer multiplication

Hardware verification - Other results

- Industrial environments: Centaur Technology
  - "A Flexible Formal Verification Framework for Industrial Scale Validation", A.Slobodova, J.Davis, S.Swords, W.Hunt, Proc. MEMOCODE'2011
    - Centaur technology: about 100 employees, designs a x86-compatible microprocessor
    - Verification: project based on open source software
    - Framework for synchronous Verilog designs, based on ACL2 + the Berkeley ZZ SAT-solver and model checker and ABC (synthesis and verification)
    - Application to floating point arithmetic circuits

Thanks for your attention

- Questions ?