Hardware Verification: Motivation

International Technology Roadmap for Semiconductors, 2005 Edition.

Verification has become the dominant cost in the design process. In current projects, verification engineers outnumber designers, with this ratio reaching two or three to one ….

...

Without major breakthroughs, verification will be a non-scalable, show-stopping barrier to further progress in the semiconductor industry.

…

The overall trend from which these breakthroughs will emerge is the shift from ad hoc verification methods to more structured, formal processes.

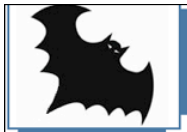
Hardware Verification Challenge

- Verification costs range from 30%-70% of the entire design cost.
- R&D for typical CPU: 500+ team, costing $0.5-1B.
- Pentium 4 (Bob Bently CAV 2005).
  - Full-chip simulation ~20Hz on Pentium 4.
  - Used ~6K CPUs 24/7: ~3 years later <1 minute of simulation cycles.
  - Exhaustive testing is impossible.
  - First large-scale formal verification at Intel: 60 person years.
  - Checked over 14K properties: Decode, Floating point, pipeline.
  - Found bugs, but no silicon bugs found to date in these areas.

Pentium FDIV (Floating point DIVision) bug in Pentium 386 led to a $475 million write-off by Intel Bob Bently CAV: $12B 2005 terms

Future of Hardware Verification

- The verification problem is getting worse.
- Nanotechnology: lots of inherently unreliable components.
- Multicores: concurrency, coherence, parallelism.
- SoC & the use of IP.
- ASICs: 18 month design cycle, $25M.
- Only 8-month selling window.
- Over 60% require respins.
- Mostly due to functional & spec errors.
  - Aart de Gues, CEO Synopsis.
- The markets are huge.
- Question: How many cell phones to be sold in 2007?



BAT

Bit-level Analysis Tool, version 0.2

Home
Papers
Download
Documentation
Examples
Benchmarks
Mailing Lists
Release Notes
License

Welcome to the homepage of BAT, the Bit-level Analysis Tool. BAT implements a state-of-the-art decision procedure for solving quantifier-free formulas over the extensional theory of fixed-size bit-vectors and fixed-size bit-vector arrays (memories). Such problems often appear in hardware, software, and security domains. BAT uses innovative techniques for efficiently translating from the high-level BAT language into tractable SAT problems. These techniques include:

- A powerful technique for greatly reducing memories.
- Subformula hashing.
- The integration of term-rewriting techniques into the translation algorithm.
- A new CNF generation algorithm.

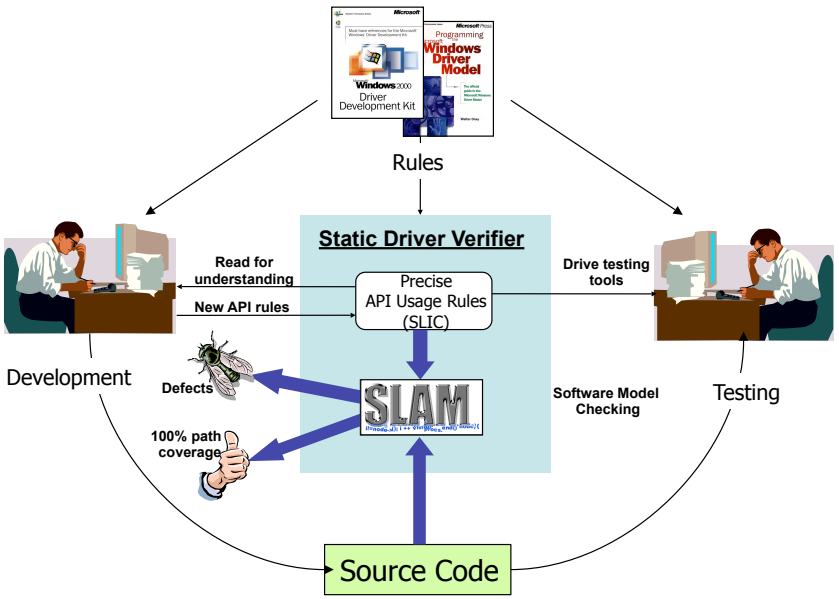
Key features of the BAT language are:

- Strongly typed, with type inference.
- Essentially equivalent to synthesizable subsets of HDLs such as Verilog.
- User-defined functions.
- Constant definitons for easily creating parameterized models.
- Memories are treated as first-class objects. This means they can be passed to functions and compared for (in)equality.

The result is a tool that can solve problems that cannot be handled by any other decision procedure we have tried. For example, BAT can prove that a **32-bit 5 stage pipelined machine model** refines its ISA **in approximately 2 minutes**. We know of no other tools that can solve even much simpler pipeline machine examples. See the benchmarks page for this and other benchmarks.

To find out more about BAT or to use it yourself, see the links along the left side of this website.

Software Verification



* Slide borrowed from SLAM Web pages

Software Verification

- "Things like even software verification, this has been the Holy Grail of computer science for many decades but now in some very key areas, for example, driver verification we're building tools that can do actual proof about the software and how it works in order to guarantee the reliability." Bill Gates, April 18, 2002. Keynote address at WinHec 2002
- Developing Drivers with the Windows® Driver Foundation, a Microsoft Press book, is now in print, including a chapter about Static Driver Verifier (SDV), which has new rules to enable analysis of drivers written against the Kernel-model Driver Framework API
- Terminator: Proving that device drivers terminate
- Many other applications: security, program analysis, ...

Programming Languages

ACL2 is ...

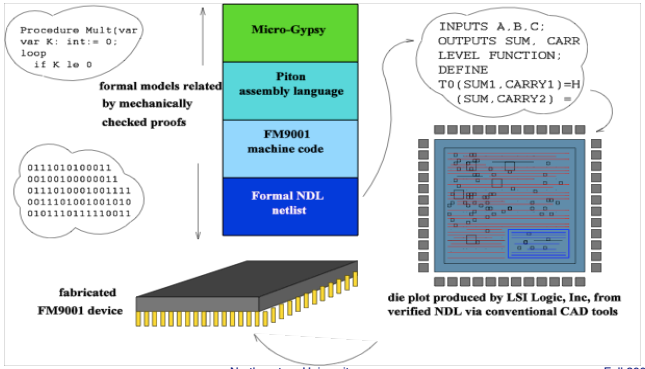
- **A programming language:**
  - Applicative, functional subset of Lisp.
  - Compilable and executable.
  - Untyped, first-order.
- **A mathematical logic:**
  - First-order predicate calculus.
  - With equality, induction, recursive definitions.
  - Ordinals up to ϵ_0 (termination & induction).
- **A mechanical theorem prover:**
  - Integrated system of ad hoc proof techniques.
  - Heavy use of term rewriting.
  - Largely written in ACL2.

ACL2

- Latest, "industrial-strength" theorem prover in the Boyer-Moore family.
- Used by AMD, Rockwell Collins, etc.
- 2005 ACM Software System Award.
- 6th workshop held with FloC.
- "A Computational Logic" for "Applicative Common LISP".
- Kaufmann & Moore.

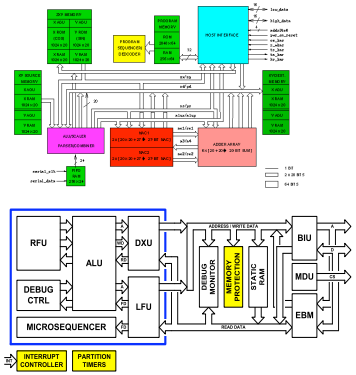
Verification with ACL2

Verification system used to prove some of the largest and most complicated theorems ever proved about commercially designed systems.



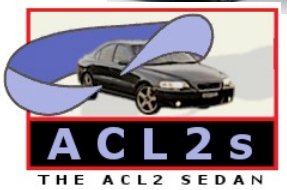
Industrial Verification with ACL2

- Motorola CAP DSP.
  - Bit/cycle-accurate model.
  - Run fasters than SPW model.
  - Proved correctness of pipeline hazard detection in microcode.
  - Verified microcode programs.
- Rockwell Collins AAMP7.
  - MILS EAL-7 certification from NSA for their crypto processor.
  - Verified separation kernel.
- Rockwell Collins JEM1.
- AMD Floating Point, … .



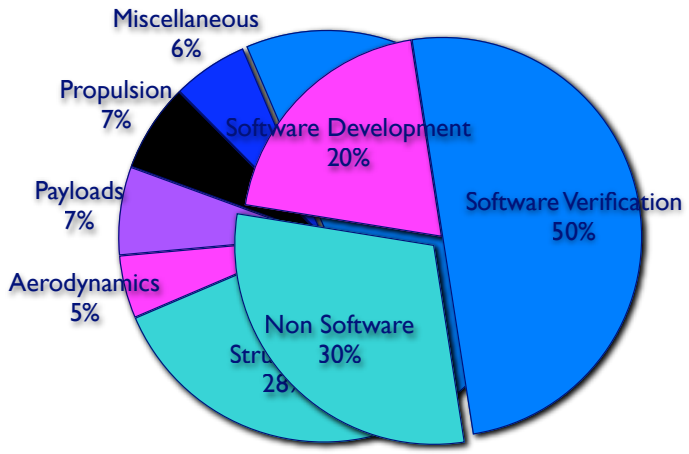
ACL2s

- ACL2 theorem prover.
  - Runs like a well-tuned race car in the hands of an expert.
  - Unfortunately, novices don't have the same experience.
  - Disseminate: wrote a book.
  - Not enough: undergrads.
- ACL2s: The ACL2 Sedan.
  - From race car to sedan.
  - Self-teaching.
  - Control a machine that is thinking about other machines.
  - Visualize what ACL2 is doing.
  - Levels & termination (FloC'06).
  - Used in several classes.
  - Available for download [DMMV'07].



Software Verification Beyond IT

Air Transport Development Costs



Is Boeing a Software Company?

- Software development and verification account for 1/3 cost.
  - Important to build reliable, dependable commercial avionics systems.
  - The industry is heavily regulated by the FAA.
- The military side is also very dependent on software.
  - 1960 - 8% F4 fighter capability came from software.
  - 2000 - 85% F22 fighter capability provided by software.
  - Even more now.

Applying Verification Technology To Other Fields

Automating the Assembly of Large-Scale Component-Based Systems

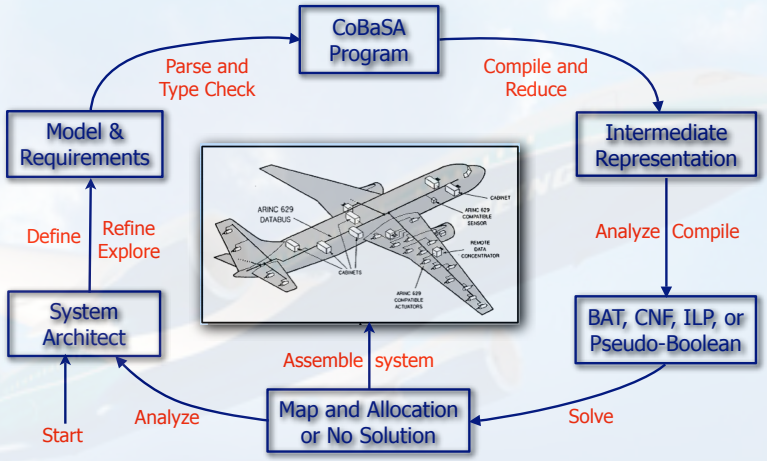
Component-Based System Design

- Goals of CBSD:
  - Construction of systems from independent components.
  - Use of commercial-off-the-shelf (COTS) components.
  - CBSD allows for separation of concerns.
  - Can decrease risk, system complexity, development time & cost.
  - Can increase reliability, malleability, and flexibility.
- Domain-specific challenges:
  - System architecture,
  - Interface definitions,
  - Trusted infrastructure,
  - Problem domain decomposition,
  - …

System Assembly

- The general challenge is the system assembly problem:
  - From a pool of available components,
  - Which should be selected &
  - How should they be connected, integrated, assembled
  - So that system requirements are satisfied?
- Currently this is application specific and labor intensive.
- Our focus is on automation:
  - Algorithmically find optimal solutions directly from requirements.
  - Insight: We can reduce system assembly to a satisfiability question.
  - Does there exist a way of selecting & assembling components that satisfies the system requirements?

Assembly of Avionics Systems



Case Study: Boeing

- The models are complex, e.g.:
  - Include I/O time, latency & network jitter.
  - Include context switching time, cache flushing time, memory latencies.
  - Based on worst-case execution time.
  - For the simplest models, it takes 1/2 month to create a CoBaSA model from a well-understood problem.
  - It takes over a man-week to check that solutions we provide.
  - Current models are over 500K in size, with thousands of constraints.
- We can solve current Boeing problems in minutes!
- Lots of interesting questions, e.g., scheduling.
- Working with NASA as well on cyber-physical verification.
  - Control Theory
  - Abstract Interpretation
  - Theorem Proving

Applying Verification Technology To Computational Biology

Pedigree Consistency

- Many problems in bioinformatics are NP-hard, e.g., pedigree consistency.
  - Relationships and genetic traits of a set of individuals
  - Check if data is consistent with the Mendelian laws of inheritance.
  - Inconsistent data can adversely affect linkage analysis (the process by which genes are linked to traits such as the predisposition to diseases).
- We solve PCC using verification technology, including BAT [MOV'07].
- Our system PCS is faster than existing algorithms, and more general.
- Actual pedigree data from a study of the genetics of grasshoper song (we are the first to solve this problem).
- Actual pedigree data on sheep from French government (INRA). We are the first to solve some of the hard problems.
- Future: Identify other hard problems in computational biology.

Syllabus

- Email addresses
- What do you want?
- Setting up Web/Wiki pages
- Will upload readings for next week
- Reschedule for Wednesdays?

SAT

Review of SAT, NP Completeness

- kSAT
  - Literals: variables or their negations
  - Clause: disjunction of literals
  - CNF formula (Conjunctive Normal Form): conjunction of clauses
  - kCNF: CNF formula w/ at most k literals per clause
  - kSAT: The set of satisfiable kCNF formulas
- Recall: SAT (= set of satisfiable CNF formulas) is NP-complete
  - NP: languages whose membership can be verified in P-time
  - NPC:
    - Hardest problems in NP
    - P-time algorithms for an NPC problem means P-time algorithm for every problem in NP
- 3SAT is NP-complete: Can reduce SAT to 3CNF ($SAT \leq_p 3CNF$)
  - Can define a P-time function f s.t. $a \in SAT$ iff $f(x) \in 3CNF$

Where Can We Draw The Line?

On the Hardness of
Satisfiability Problems

*Blue slides mostly borrowed from D. Moshkovits

Introduction

- Objectives:
  - To show variants of SAT and check if they are NP-hard
- Overview:
  - Known results
  - 2SAT
  - Max2SAT

What Do We Know?

- Checking if a propositional calculus formula is satisfiable (SAT) is NP-hard.

Example: propositional calculus formula

$\neg(x \wedge \neg z \wedge (\neg w \vee x)) \vee (x \wedge \neg y) \rightarrow \neg y$

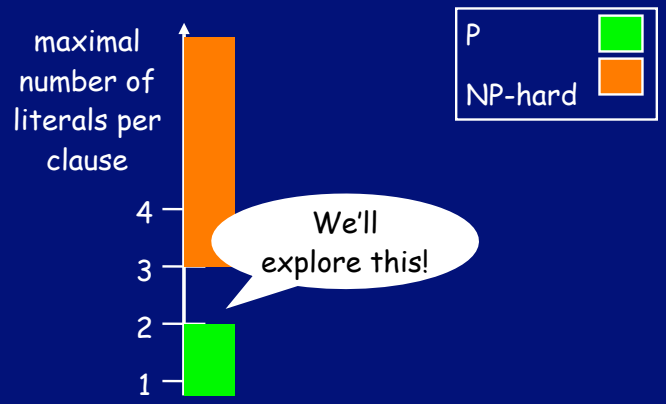
What Do We Know?

- We concentrated on a special case: CNF formulas.

structure of CNF formulas

$(..\vee..\vee.....\vee..) \wedge ... \wedge (..\vee..\vee.....\vee..)$

What Do We Know?



2SAT

- Instance: A 2-CNF formula φ
- Problem: To decide if φ is satisfiable

Example: a 2CNF formula

$(\neg x \vee y) \wedge (\neg y \vee z) \wedge (x \vee \neg z) \wedge (z \vee y)$

2SAT is in P

Theorem: 2SAT is polynomial-time decidable.

Proof: We'll show how to solve this problem efficiently using path searches in graphs…

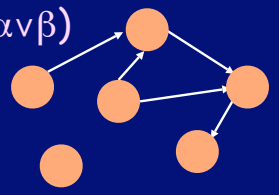
Searching in Graphs

Theorem: Given a graph $G=(V,E)$ and two vertices $s,t \in V$, finding if there is a path from s to t in G is polynomial-time decidable.

Proof: Use some search algorithm (DFS/BFS). ■

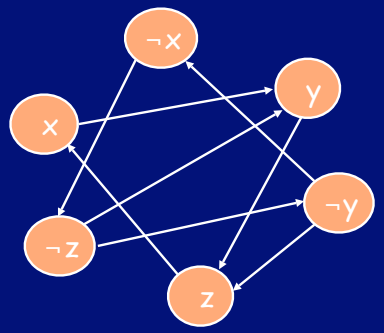
Graph Construction

- Vertex for each variable and a negation of a variable
- Edge (α,β) iff there exists a clause equivalent to $(\neg\alpha \vee \beta)$



Graph Construction: Example

$(\neg x \vee y) \wedge (\neg y \vee z) \wedge (x \vee \neg z) \wedge (z \vee y)$



Observation

Claim: If the graph contains a path from α to β, it also contains a path from $\neg\beta$ to $\neg\alpha$.

Proof: If there's an edge (α,β), then there's also an edge $(\neg\beta,\neg\alpha)$.

Correctness

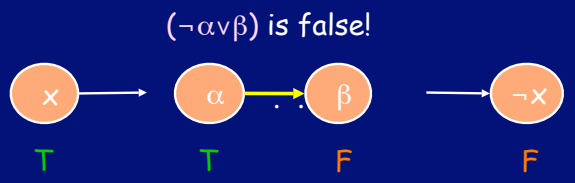
Claim:

a 2-CNF formula φ is unsatisfiable iff there exists a variable x, such that:

1. there is a path from x to $\neg$x in the graph
2. there is a path from $\neg$x to x in the graph

Correctness (1)

- Suppose there are paths x..¬x and ¬x..x for some variable x, but there's also a satisfying assignment ρ.
- If ρ(x)=T (similarly for ρ(x)=F):

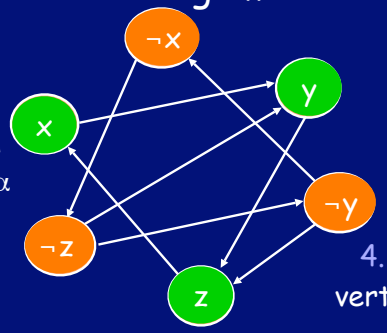


Correctness (2)

(¬x∨y)∧(¬y∨z)∧(x∨¬z)∧(z∨y)

- Suppose there are no such paths.
- Construct an assignment as follows:

1. pick an unassigned literal α, with no path from α to ¬α, and assign it T



2. assign T to all reachable vertices

3. assign F to their negations

4. Repeat until all vertices are assigned

Correctness (2)

Claim: The algorithm is well defined.

Proof: If there were a path from x to both y and ¬y,

then there would have been a path from x to ¬y and from ¬y to ¬x.

Correctness

A formula is unsatisfiable iff there are no paths of the form x..¬x and ¬x..x. ■

2SAT is in P

We get the following efficient algorithm for 2SAT:

- For each variable x find if there is a path from x to $\neg x$ and vice-versa.
- Reject if any of these tests succeeded.
- Accept otherwise

$\Rightarrow$ 2SAT$\in$P. ■

Max2SAT

- Instance: A 2-CNF formula φ and a goal K.
- Problem: To decide if there is an assignment satisfying at least K of φ's clauses.

Example: a 2CNF formula

$(\neg x \vee y) \wedge$
$(\neg y \vee z) \wedge$
$(x \vee \neg z) \wedge$
$(z \vee y)$

Max2SAT is in NPC

Theorem: Max2SAT is NP-Complete.

Proof: Max2SAT is clearly in NP.

We'll show 3SAT$\leq_p$Max2SAT.

Gadgets

Claim: Let

$\psi(x,y,z,w) = (x)\wedge(y)\wedge(z)\wedge(w)\wedge$
$(\neg x\vee\neg y)\wedge(\neg y\vee\neg z)\wedge(\neg z\vee\neg x)\wedge$
$(x\vee\neg w)\wedge(y\vee\neg w)\wedge(z\vee\neg w).$

Proof: By checking.

- Every satisfying assignment for $(x\vee y\vee z)$ can be extended into an assignment that satisfies exactly 7 of the clauses.
- Other assignments can satisfy at most 6 of the clauses.

The Construction

- For each $1 \le i \le m$, replace the i-th clause of the 3-CNF formula $(\alpha \vee \beta \vee \gamma)$ with a corresponding $\psi(\alpha,\beta,\gamma,w_i)$ to get a 2-CNF formula.
- Fix K=7m.

Make sure this construction is poly-time

Correctness

- Every satisfying assignment for the 3-CNF formula can be extended into an assignment that satisfies 7m clauses.
- If 7m clauses of the 2-CNF formula are satisfied, each ψ has 7 satisfied clauses, so the original formula is satisfied.

Corollary

$\Rightarrow 3SAT \leq_p Max2SAT$ and $Max2SAT \in NP$

$\Rightarrow$ Max2SAT is NP-Complete. ■

Summary

- We've seen that checking if a given CNF formula is satisfiable is:
  - Polynomial-time decidable, if every clause contains up to 2 literals.
  - NP-hard, if each clause may contain more than 2 literals.
- We've also seen Max2SAT is NP-hard.

Conclusions

- A special case of a NP-hard problem may be polynomial time decidable.
- The optimization version of a polynomial-time decidable problem may be NP-hard.
- Questions:
  - Horn clause: at most one positive literal.
  - Examples: $(\neg x \vee y)$, $(\neg x \vee \neg y \vee \neg z)$, (x).
  - Is HORNSAT in P? NPC?