An Application in Detail: Boolean Satisfiability in Configurable Hardware

- Combinatorial problem common in CAD, AI domains
  - Automatic test pattern generation
  - Logic synthesis
  - Commercially-relevant data sets take hours or days to solve
- Definition: Given a Boolean formula, find an assignment to the variables such that formula evaluates to true, OR prove no such assignment exists.
  - Input normally in conjunctive normal form (CNF)
  - aka product-of-sum form
Basic SAT Algorithm

- Davis-Putnam Algorithm
  - Backtrack search

- Algorithm Steps:
  - Assign 1 variable at a time
  - For each new assignment, compute transitive implications
  - Check for contradictions and backtrack if needed

Diagram:

```
        Start
       /    \
a = 0  a = 1
  /    /    \    \    \    \    
b = 0 b = 1 b = 0 b = 1
```

Starting state: a = 0, b = 0

Options:
- a = 1, b = 0
- a = 1, b = 1
- a = 0, b = 1
2 Brief Examples

Formula: $(a+c')(a+d')(a+b'+c)(a'+e)(a'+c)$

- **Step 1**: Set $a=0$:
  - implies $c=0$
  - implies $d=0$
  - $c$ implied to $0$ implies $b=0$
  - No contradictions.

- **Step 2**: Set $e=0$
  - Solution found!

Formula: $(a+b)(c+d)(a+b')$

- **Step 1**: Set $a=0$:
  - implies $b=1$
  - $b=1$ implies $a=1$!
  - Contradiction!

- **Step 2**: Set $a=1$: no impl.

- **Step 3**: Set $b=0$: no impl.

- **Step 4**: Set $c=0$:
  - implies $d=1$
  - Solution found!
Mapping SAT to Hardware

- Template design customized for each formula
- Per-variable state-machines manage assignments
- Linear array of state machines for tree search
  - Only one variable is in active control at a time
- Speedup:
  - Implications mapped directly to gates
  - Parallelism in implication processing
Implication Circuit

Part of solver circuit for formula:

\[(a + c')(a + d')(a + b' + c)(a' + e)(a' + c)\]
Cell contains implication ckt and state machine
We built it ... Twice!

- Initially on: PCI PAMette
  - small circuits
  - proof-of-concept

- And then on: Ikos logic emulator
  - Virtual Wires: Easier to compile, partition
  - 64 XC4013 FPGA chips per emulator board

- Currently have full-sized (DIMACS) benchmarks running.
Problem Solving Procedure:

- Fully-automated path from Boolean formula to a solution
- Will invoke on tough problems as part of larger ATPG software package
Evaluation Methodology

- DIMACS Challenge and UCSC benchmarks
- Compared against GRASP using comparable algorithm
  - Speed-up ratio expressed in histogram
- Use C simulator to measure solution time cycle count
  - VHDL simulation too slow
  - Real Ikos runs difficult to profile
- 1.33 MHz cycle time
  - chosen via Ikos compiles
Performance

- 45% of problems have > 100X run-time speedup
- Speedups do not include compile-time
Discussion

Advantages of Formula-Specific SAT solver

- Customized connections avoid memory bottleneck
- Formula-specific design achieves high, fine-grain parallelism

Disadvantages of Formula-Specific SAT solver

- Some irregular wiring makes designs hard to route
- Difficulty in partitioning can degrade clock rates
- Compile-time is part of timing for each run

Our Response?
SAT-II: A Clause-based Modular Design

- Each FPGA holds a number of clause-boxes
- Each clause-box handles implications for one clause
- Pipelined bus allows higher clock rate (20MHz)
- Addresses disadvantages from prior solver
  - Wiring: much more regular
  - Partitioning: done as part of template design
  - Compile-time: done once per basic design and specialized for each formula (much faster)
Organization of Pipelined Bus

- Each FPGA one stage of pipelined bus
- One cell per SAT clause
  - Set variable values on bus
  - Bus decoder
  - Implication circuit
  - Analysis circuit
- Data values updated around ring
- Preliminary results: 10-1000X faster than SAT-I
Configurable Hardware Summary

- Customization of SAT problem to configurable hardware leads to impressive speedups
  - 1-5 orders of magnitude
- Complex, control-intensive applications can be implemented in configurable hardware.
- Synthesis: Connection of profile system to synthesis system currently in progress
  - Profiler tools feed off application experiences
  - And, vice versa.