Boolean Satisfiability in Verification

Gianpiero Cabodi *Stefano Quer*

Politecnico di Torino

Torino, Italy

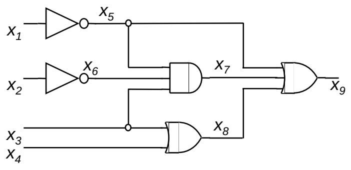
{gianpiero.cabodi,stefano.quer}@polito.it

http://staff.polito.it/{gianpiero.cabodi,stefano.quer}/

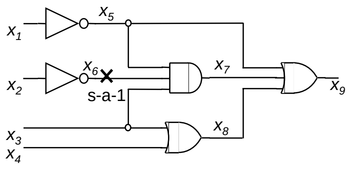
Applications of SAT in EDA

- ❖ Test Pattern Generation:
  - ◆ Stuck-at, Delay faults, etc.
  - ◆ Redundancy Removal
- ❖ Circuit Delay Computation
- ❖ Combinational Equivalence Checking
- ❖ Bounded/Unbounded Model Checking
- ❖ Superscalar processor verification
- ❖ FPGA routing
- ❖ Noise analysis

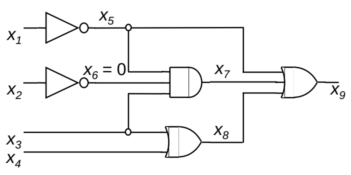
ATPG



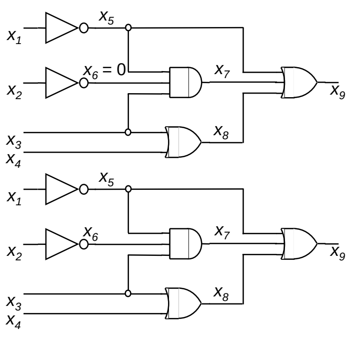
ATPG



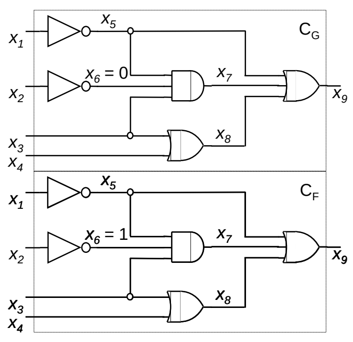
ATPG



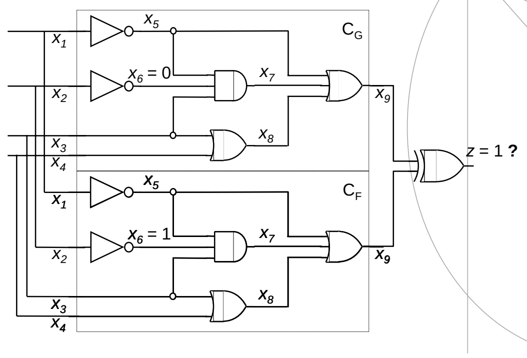
ATPG



ATPG



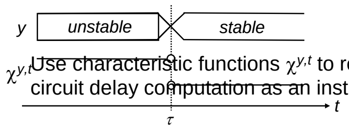
ATPG



Delay Computation Using SAT

Can circuit delay be $\geq \Delta$?

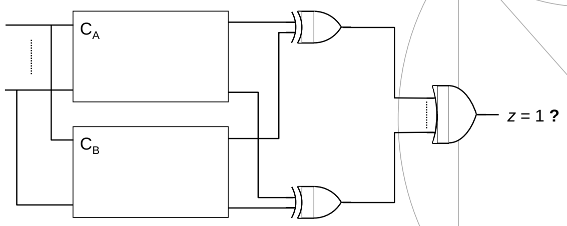
Characteristic Function [McGeer,ICCAD'91]



Use characteristic functions $\chi^{y,t}$ to represent circuit delay computation as an instance of SAT **!**

$\chi^{y,t} = 1 \Leftrightarrow$ node y stabilizes no ealier than t

Combinational Equivalence Checking



If $z = 1$ is unsatisfiable, the two circuits are equivalent **!**

Bounded Model Checking (BMC)

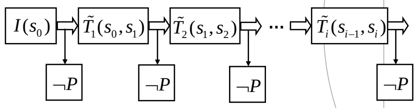
- ❖ **Bounded Model Checking (Biere, et al., TACAS 1999)**
  - ◆ **Property checking method based on finite unfolding of transition relation interleaved with checks of the property**
    - ✧**Sound – in its pure form no false positives are possible**
    - ✧**Incomplete – cannot guarantee correctness of property**
  - ◆ **Basic method**
    - ✧**CNF-based**
      - – Use CNF-based SAT solver to represent unfolding and proof UNSAT for correctness of property
    - ✧**Circuit-based**
      - – Use ATPG-like reasoning to show untestability
    - ✧**Hybrid**
      - – Use circuit rewriting and SAT checking interleaved
        - • e.g. based on AND/INV graphs

- ❖ **Given**
  - ◆ **A finite transition system M**
  - ◆ **A property P (representing "good" states)**
- ❖ **Note: We restrict our attention to safety properties**
- ❖ **Does M allow a counterexample to ¬P of *k transitions or fewer*?**

BMC Unfolding

- Property ***P*** holds in states k following initial state I_0

  $S_0 \wedge$
  $TR(S_0,S_1) \wedge TR(S_1,S_2) \wedge \dots \wedge TR(S_{k-1},S_k) \wedge$
  $\neg P(S_k)$

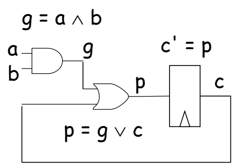


- This problem can be translated into a SAT problem
- Create an instance of SAT
  - CNF format
  - A counterexample is a path from a state satisfying S_0 to state satisfying P, where every transition satisfies TR
- Bounded Model Check for length *k*

```
Algorithm BMC(max_length){
    forall i = 1 and  i < max_length do {
      if (SAT(BMC_i)) return FAIL
    }
    return SUCCESS;
}
```

Example

Transition system described by a set of constraints



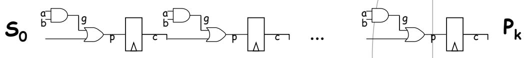
Model:

C = {
  g = a ∧ b,
  p = g ∨ c,
  c' = p
}

Each circuit element is a constraint
  note: $a = a_t$ and $a' = a_{t+1}$

- **Unfold the model k times**

  $U_k = TR_0 \wedge TR_1 \wedge \dots \wedge TR_{k-1}$



- Use SAT solver to check satisfiability of

  $S_0 \;\wedge\; U_k \;\wedge\; P_k$

- *A satisfying assignment is a counterexample of k steps*

Applications

- Debugging
  - Can find counterexamples using a SAT solver
- Proving properties
  - Only possible if a bound on the length of the shortest counterexample is known
  - I.e., we need a *diameter* bound. The diameter is the maximum length of the shortest path between any two states
  - Worst case is exponential. Obtaining better bounds is sometimes possible, but generally intractable

Unbounded Model Checking (UMC)

- We consider a variety of methods to exploit SAT and BMC for unbounded model checking
  - K-step induction
  - Abstraction
    - ✧Counterexample-based
    - ✧Non-counterexample-based
  - Exact image computations
    - ✧SAT solver tests for fixed point
    - ✧SAT solver computes image
  - Over-approximate image computations

Improvements *(Sheeran, FMCAD 2000)*

- **Assert correctness of properties proven for previous frames** $tp^k(s_0,s_k) = \bigwedge_{0 \le i<k} p(s_i) \wedge t(s_i,s_{i+1})$
  - Helps pruning the search, especially for optimization in this talk
- Simple paths constraints
  - Do not allow that a state is visited twice

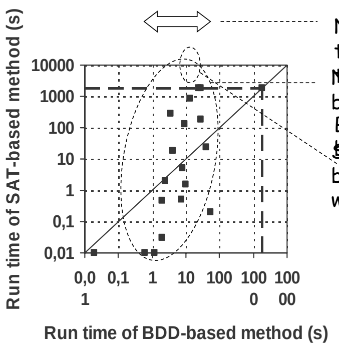
$$tp^k_{simple}(s_0,s_k) = \bigwedge_{0 \le i<k} p(s_i) \wedge t(s_i,s_{i+1}) \wedge \bigwedge_{0 \le i<j \le k} s_i \neq s_j$$

  - Does not really help in practice
- *K*-step inductiveness:
  - In addition to *BMC*$_k$ check also

$$inv^k = tp^k(s_0,s_k) \wedge \neg p(s_k)$$

  - Makes proof complete

SAT versus BDDs (McMillan, CAV 2002)



Note low variance in times for BDD based technique.

Note low correlation between the two methods.

Benchmarks may be biased in favor of BDD's.

BDD's are better overall.

SAT based method may be a good alternative when BDD's fail.

But note relative immaturity of SAT based method

Context

- **SAT is the quintessential NP-complete problem**
- **Theoretically well-studied**
- **Practical algorithms for large problem instances started emerging in the last five years**
- **Has many applications in EDA and other fields**
- **Can potentially have similar impact on EDA as BDDs**
- **EDA professionals should have good working knowledge of SAT formulations and algorithms**

Research Directions

- **Algorithms**
  - **Explore relation between different techniques**
    - **backtrack search; conflict analysis; recursive learning; branch-merge rule; randomization & restarts; clause inference; local search (?); BDDs (?)**
  - **Address specific solvers (circuits, incremental, etc.)**
  - **Develop visualization aids for helping to better understand problem hardness**
- **Applications**
  - **Industry has applied SAT solvers to different applications**
    - **SAT research requires challenging and representative publicly available benchmark instances** !

Conclusion

- **SAT solvers are very effective at ignoring irrelevant facts**
- **SAT solvers can produce refutations**
- **We can exploit in a number of ways**
  - **BMC**
  - **Abstraction for UMC**
  - **Abstract image computations using interpolation**

This makes it possible to model check *localizable* properties large systems

- **Approaches that compute exact images sacrifice this quality of SAT solvers**
  - **still useful as alternative to BDD's**
- **For non-localizable properties, SAT-based BMC and UMC do not perform well**
- **The capacity of SAT-based UMC is smaller than the one of BMC**
  - **Need to settle for bounded results**
  - **Debugging solution instead of complete verification**
  - **Use UMC only in late verification phases**