Reusing Learned Information in SAT-based ATPG

Görschwin Fey, Tim Warrode, Rolf Drechsler
University of Bremen
drechsle@informatik.uni-bremen.de

Work supported in part by PHILIPS and DFG
Outline

- Introduction/Motivation
- Preliminaries
  - Circuit, Fault Model, Test Pattern Generation
- Proof techniques
  - Boolean satisfiability, SAT, Circuit to SAT Conversion
- SAT-based ATPG
  - Problem Description
  - Learning
- Experimental Results
- Conclusions
Motivation

• Increasing size of circuits

• Post-production test is a crucial step:
  – Have there been problems during production?
  – Does the circuit contain faults?

• Test patterns are applied
Motivation

- Test pattern generation happens at the Boolean level

- Classical ATPG algorithms reach their limits

➤ There is a need for more efficient ATPG tools!
Fault Model

• Model “realistic” fault
  – Physical faults or defects at the Boolean level
• Simplified assumption
• Based on netlist

• Static or dynamic
  – Here: static only
Stuck-at Fault Model

- Single line is assumed to have a fixed value (0 or 1)

- Example: stuck-at 0 fault at line d
  
  ![Correct Circuit](image1)
  ![Faulty Circuit](image2)
Test Pattern Generation

- Physical defects are modeled on the Boolean level

![Stuck-at-0 Diagram]

- Automatic Test Pattern Generation (ATPG)
  Given: Circuit C and fault model F
  Objective: Calculate test patterns for faults in C with respect to F
Boolean Difference

- BD of faulty and fault free circuit
Fault Classification

• If there is a test, the fault is testable.
• If there does not exist a test, the fault is redundant.

• Decision is NP complete.
General Structure

- Justification and propagation
Improvements

- D-algorithm
- PODEM: Only branch on inputs
- FAN: Branching on fanout stems
- SOCRATES: Learning
- HANIBAL: Recursive learning

- Alternative: SAT-based
  - Formulation based on formal techniques
  - Proof techniques: SAT
SAT

- A single test-vector is sufficient
- Construction of satisfying assignment

**SAT-problem:** For a given Boolean function $f$ find an assignment $a$, such that $f(a) = 1$ or prove that such an assignment does not exist.
CNF for Circuit

\[ \varphi = h \left[ d = (ab) \right] \left[ e = \neg(b + c) \right] \left[ f = \neg d \right] \left[ g = d + e \right] \left[ h = fg \right] \]

\[ = h \]

\[ (a + d)(b + d)(\neg a + \neg b + \neg d) \]
\[ (\neg b + \neg e)(\neg c + \neg e)(b + c + e) \]
\[ (\neg d + \neg f)(d + f) \]
\[ (\neg d + g)(\neg e + g)(d + e + \neg g) \]
\[ (f + \neg h)(g + \neg h)(\neg f + \neg g + h) \]

- CNF for circuit and assignment \( h=1 \)
SAT Solving

- Most algorithms are based on DLL procedure

- Overall flow
  - Assign variables in the CNF
  - If a contradiction occurs backtrack
Basic Procedure

start

Implications (BCP)

All clauses satisfied?

Formula satisfied

Yes

contradiction?

No

Assign next variable based on heuristics

No

Do there exist alternatives?

Yes

Alternative assignments according to tree structure

No

Formula cannot be satisfied
Implications

• Unit clause: Only one unspecified literal

\[ \neg a + b + \neg c \]

\[
\begin{array}{c|c}
1 & 0 \\
\end{array}
\]

\[ \Rightarrow c = 0 \]

• Boolean Constraint Propagation (BCP) is based on iteration of unit clause rule

• BCP corresponds to implications on the net list

• Fast implementation, since CNF is very regular
Motivation for SAT-based ATPG

- Substantial improvements in SAT solving

Use
- Advanced SAT techniques
- In combination with structural information

For
- Large industrial circuits
- In a multi-valued domain
Test Pattern Generation

CORRECT

FAULTY

$\chi$

Stuck-at-0

Inputs

Outputs

Output

Output

BD $\neq 1$
SAT-based ATPG

- **Input:** Circuit $C$, fault $F$

  1. Fault modeling:
     BD between fault free and faulty circuit
  2. Translate into CNF
  3. Use SAT solver to calculate solution

- **Output:** Classification of $F$, test vector $T$
Use of Structural Information

- Influenced circuit parts
Create Instance

- Build circuit structure accordingly
Fault Modeling
CNF

- \( F = (c + \bar{d}_g + e_g) \cdot (c + \bar{e}_g) \cdot (d_g + \bar{e}_g) \)
- \( \cdot (a + b + \bar{d}_g) \cdot (\bar{a} + d_g) \cdot (\bar{b} + d_g) \cdot (d_f) \)
- \( \cdot (c + \bar{d}_f + e_f) \cdot (c + e_f) \cdot (d_f + \bar{e}_f) \)
- \( \cdot (e_g + e_f + BD) \cdot (\bar{e}_g + \bar{e}_f + BD) \)
- \( \cdot (\bar{e}_g + e_f + BD) \cdot (e_g + \bar{e}_f + BD) \cdot (BD) \)

- \( F \) is the CNF for circuit with \( d \) s-a-1
- Inputs satisfy CNF \( \rightarrow \) can detect fault
- CNF is linear in circuit size
Features of PASSAT

- Structural information (cf. TEGUS)
- Memory management
- Advanced SAT techniques
- Problem specific variable selection
- Multi-valued model
Advanced SAT Techniques

• Built-in techniques from Zchaff
  – Conflict based learning
  – Non-chronological backtracking
  – Event-driven evaluations
  – Clever decision heuristics
Learning Techniques

- **Incremental SAT**
  - Modify instance on the fly
  - Learned information in kept
  - Similar faults have similar conflicts in the SAT instance
  - How many clauses to learn?
    - Heuristics

- Learning: static and dynamic
Incremental SAT
Industrial Circuits – Number of Aborted Faults

![Bar chart showing number of aborted faults for different circuits. The chart labels the circuits as p177k, p462k, p565k, and p1330k, and differentiates between 'classic' and 'heu' faults with colored bars.]
Industrial Circuits – Run Time

![Bar chart showing run times for p177k, p462k, p565k, and p1330k with 'classic' and 'heu' categories.]
Conclusions

- SAT for ATPG
- Formulation based on formal techniques
- Use of learning techniques
- Improvements of up to a factor of 5

- Better run times for “hard” faults
- Applicable to large industrial circuits