Accelerate SAT-based ATPG via Preprocessing and New Conflict Management Heuristics

Junhua Huang Hui-Ling Zhen Naixing Wang

Mingxuan Yuan Hui Mao Yu Huang Jiping Tao

Outline

- Introduction of SAT-based ATPG
- Research Background on SAT-based ATPG
- Our New Framework
- Experimental results
- Conclusions

Outline

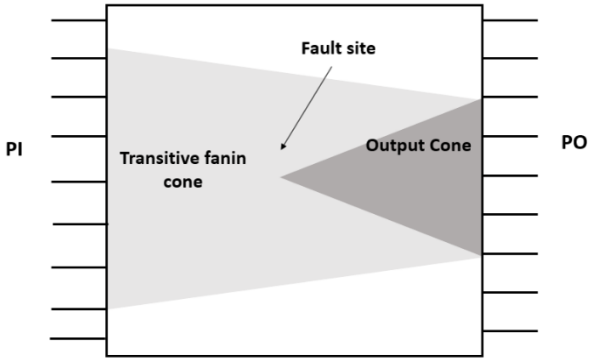
- Introduction of SAT-based ATPG
- Research Background on SAT-based ATPG
- Our New Framework
- Experimental results
- Conclusions

Introduction of SAT-based ATPG

- Several structural heuristics have been proposed for ATPG scenarios:
  - For instance, D-algorithm, PODEM, etc.
  - Core idea: Brach and bound algorithm
  - Main procedure:
    ① Activate the fault
    ② Forward propagation
    ③ Perform backward implication to satisfy J-frontier
    ④ Check consistency, if not, backtrack and re-make decision
    ⑤ Satisfy all objectives and return the test pattern, or prove untestable

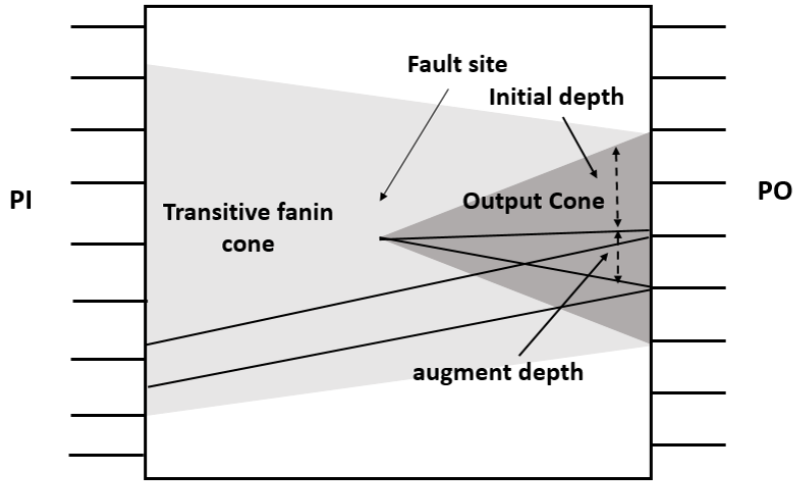
Introduction of SAT-based ATPG

- Boolean Satisfactory (SAT) is a robust alternative for conventional algorithms:
  - There have also been many SAT engines.
    - PASSAT, TG-Pro, TI-GUAN, etc.
  - Two main steps in SAT-based ATPG
    - Circuit-to-CNF transformation:
      - Conjunctive Normal Form (CNF) is the basic format of modern SAT solver
      - Two parts of CNF: circuit part & fault information (e.g., D-chain path)
    - Solve the corresponding CNF expression
  - Solving SAT is also a branch-and-bound algorithm.
  - Modern SAT solver is also conflict-driven.

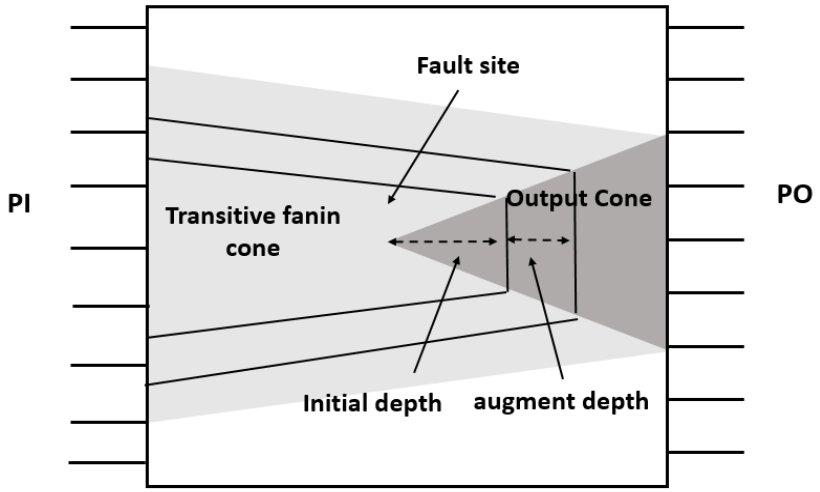


Introduction of SAT-based ATPG

- Several Incremental algorithms have been proposed for CNF generation.
- SAT is based on symbolic computation, solution of part CNF can help.



Incremental SAT-based Engine.
A kind of incremental Fanin method.
Finding a solution is enough for SAT instance



Incremental UNSAT-based Engine.
A kind of incremental Fanout method.
More and more quickly to find conflicts.

Outline

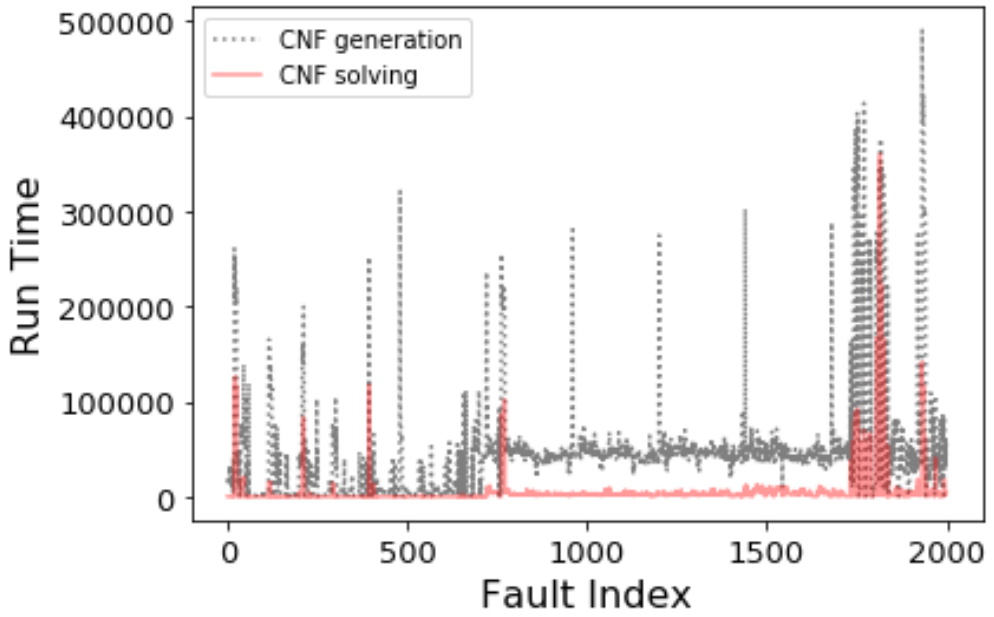
- Introduction of SAT-based ATPG
- Research Background on SAT-based ATPG
- Our New Framework
- Experimental results
- Conclusions

Research Background on SAT-based ATPG

- Recent SAT-based ATPG works are ALL on transformation methods.
  - Their core idea is to embed more and more fault information in CNF
  - Reason:
    - ① SAT formula losses the structural information.
    - ② More information can improve solving efficiency.
    - ③ SAT solver can be more and more quickly in modern days.
- What is the real bottleneck for SAT applying for industrial ATPG?

Research Background on SAT-based ATPG

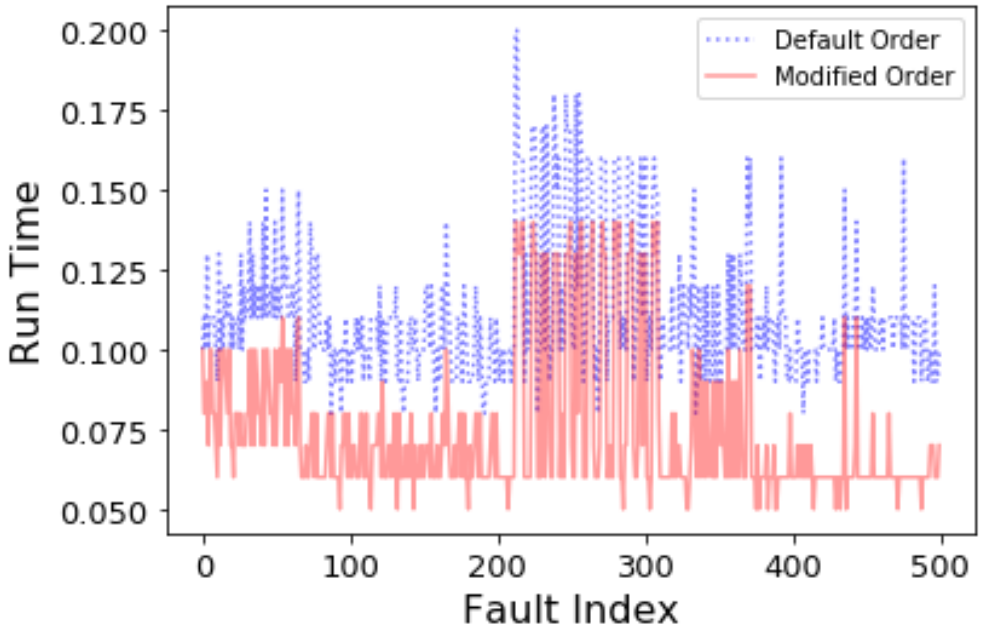
- Bottleneck 1: In many cases, Circuit-to-CNF transformation time is much larger than solving.
- Taking one industrial circuit as an example, the comparison on generation time and solving time is below.
- It is found that generation time is really much more larger than solving.
- Decreasing the generation time is real bottleneck.



Horizontal axis: fault index
Vertex axis: generation or solving time
(Unit: microseconds)

Research Background on SAT-based ATPG

- Bottleneck 2: Generation and solving are two independent modules.
- Solving friendly? Not only adding clauses.
- Examples: If we can have a better order of different variables, solving performance can be improved.
- Solving friendly may also means for better rank, better decision, better etc.

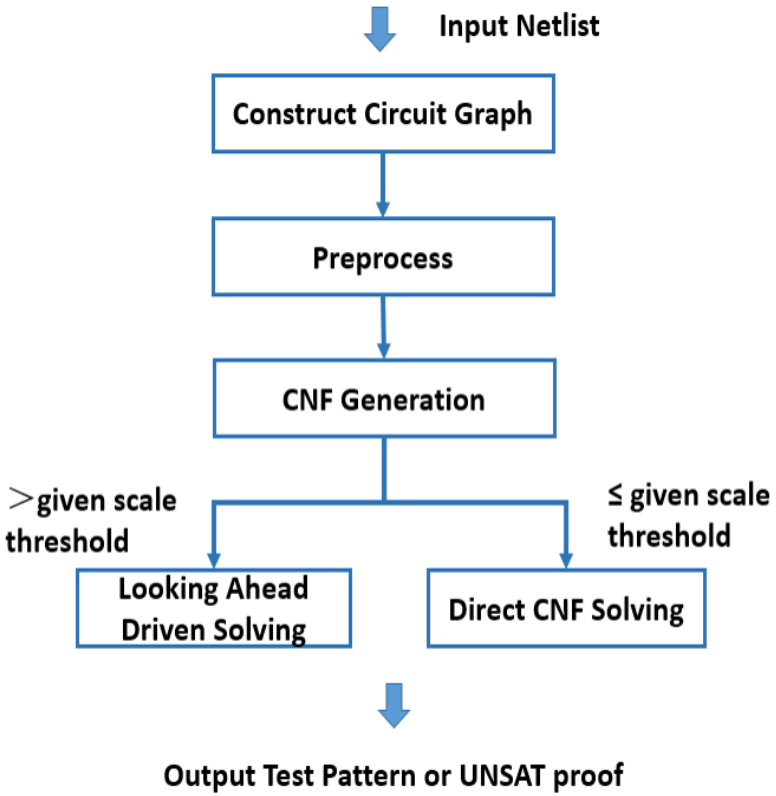


Horizontal axis: fault index
Vertex axis: solving time
(Unit: seconds)

Outline

- Introduction of SAT-based ATPG
- Research Background on SAT-based ATPG
- Our New Framework
- Experimental results
- Conclusions

Our New Framework



① Input netlist and construct circuit graph

- Circuit graph is a DAG, paving for the following graph algorithm

② A new preprocessing algorithm

- For simplification

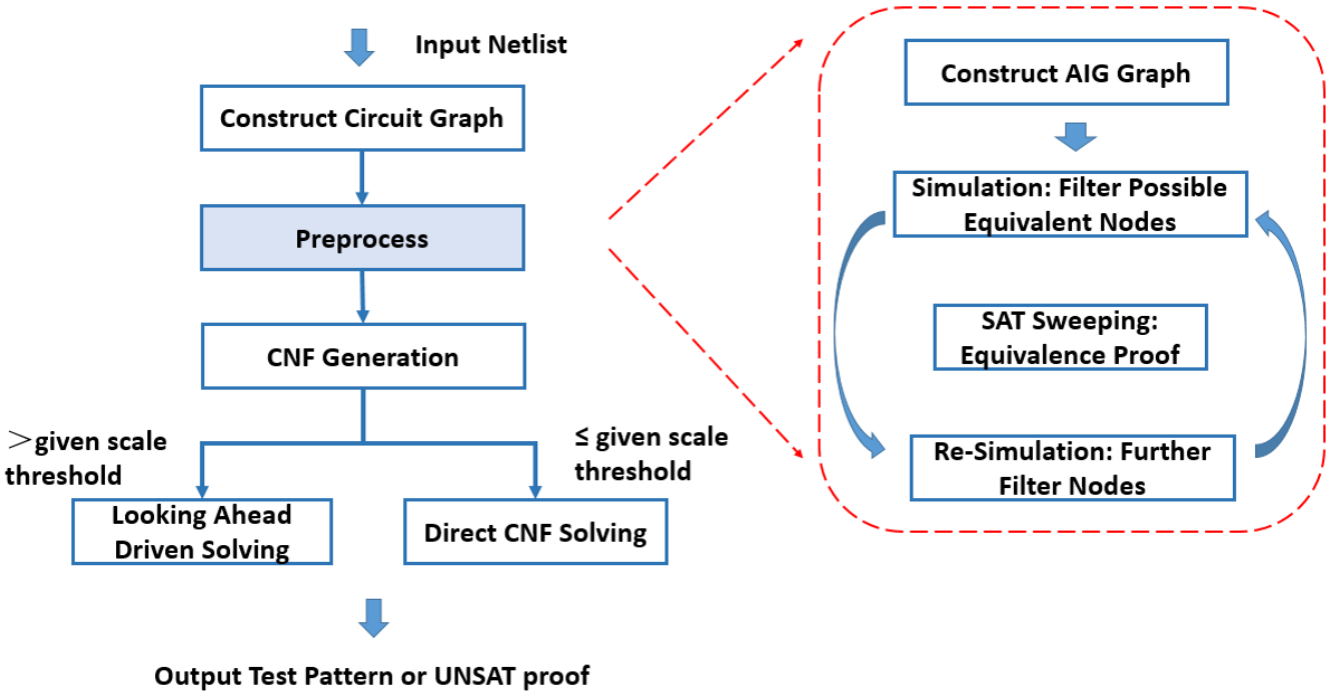
③ Ordered Incremental Circuit-to-CNF generation

- Ranking gates can help to be solving friendly.

④ Different solving methods for CNF solving.

- Less on the threshold: Solving directly.
- Larger than the threshold: Cube and Conquer solving method.

Our New Framework



Three main steps for preprocessing:

① Simulation:

a) Random patterns are for PIs in AIG.

b) If their outputs are same, there is a higher probability to be equal

② SAT sweeping.

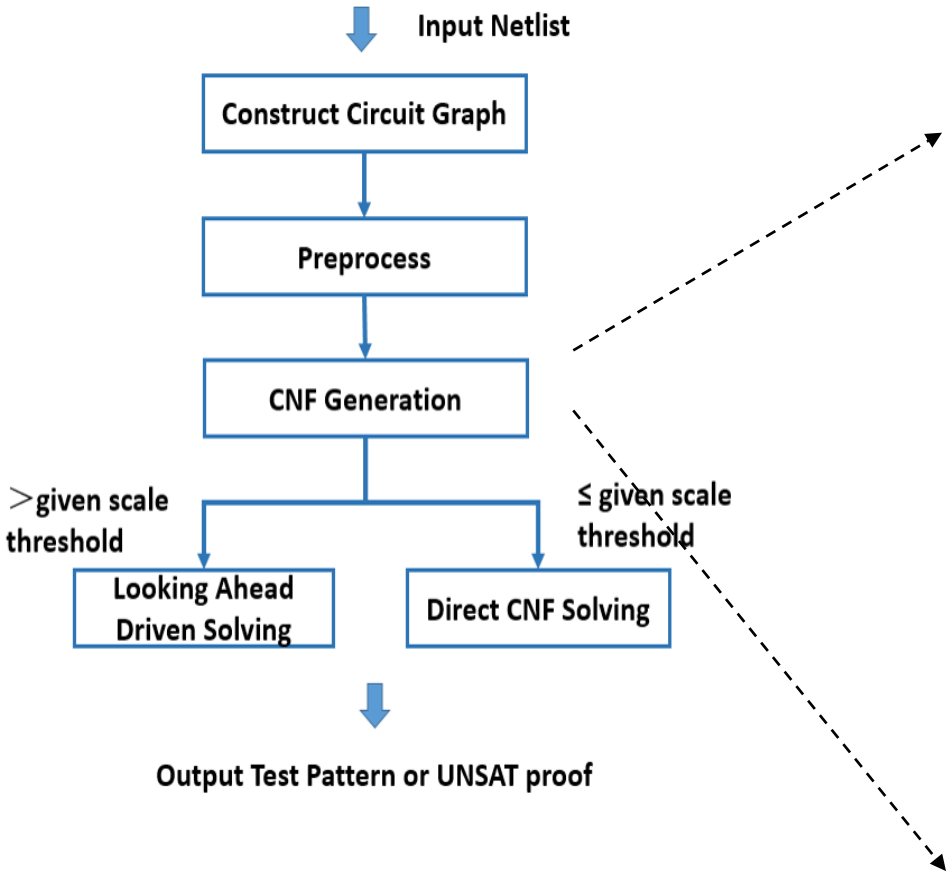
a) Prove equality.

b) If so, merge the nodes for simplification

③ Re-simulation for following decreasing.

④ Hold on the loop, until reach the threshold.

Our New Framework



Incremental CNF generation way, and the difference is in ordering gates:

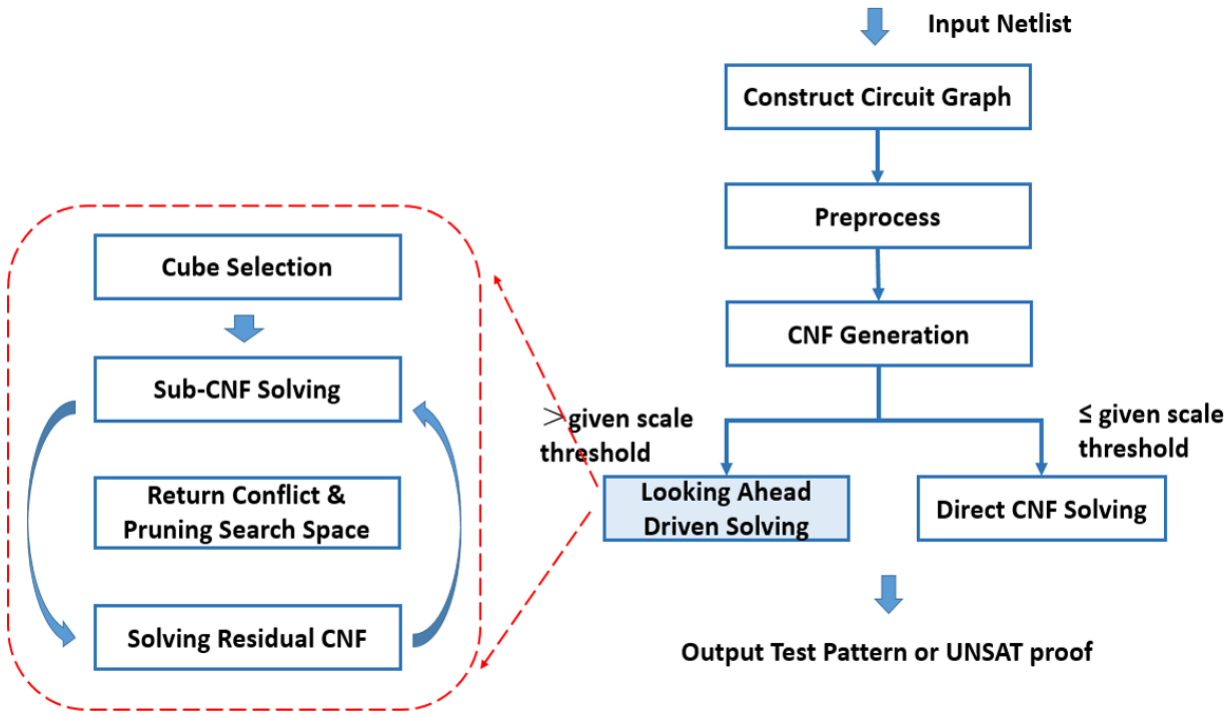
① Initialization with a doubly linked list, in which the initial literal ordering is consistent with the pin ordering during fault propagation from PIs to POs. Each variable has an initial score.

② We increases the score of a variable incrementally by 1 when it appears in a newly learnt conflict.

③ The core is decreased periodically according to the decision level of learnt conflicts

④ All literals are ranked via the scores, and the decision tree would be branched with the largest score.

Our New Framework



Three steps for relatively large instance:

① Partition CNF into different sub-CNFs, each of which corresponds to a FFR as well as its related logic cone in fault propagation.

② Solving cubes and return conflicts for looking-ahead.

a) One or several sub-CNFs are first solved.

b) If one of the sub-CNF is UNSAT, we will stop the computation and the fault is untestable.

c) Otherwise, it would return partial assignments and learnt conflicts.

③ Solve the residual CNF based partial assignment and learnt conflicts.

Outline

- Introduction of SAT-based ATPG
- Research Background on SAT-based ATPG
- Our New Framework
- Experimental results
- Conclusions

Experimental Setup

- Benchmarks: Hisilicon industrial circuits
- Baselines:
  a) Original SAT-based ATPG engine: TG-Pro.
  b) Original Incremental SAT-based ATPG engine.
  c) Commercial tool: structural heuristics
- Hard-to-detect stuck-at faults and leading-edge sequential faults.
  a) First round: Given by structural heuristics, given backtrack limits
  b) Second round: Different SAT-based methods.

Experimental Results

TABLE I
EVALUATION ON RUNTIME (SECONDS)

Circuit	#gates	TG-Pro	Incre TG-Pro	Commercial	New Approach
Case1	4891k	2943.8	1107.4	264.5	**140.9**
Case2	221k	28.7	15.7	15.1	**3.4**
Case3	206k	24.6	9.4	3.3	**2.8**
Case4	246k	35.8	15.1	3.44	4.1
Case5	785k	276.4	105.9	83.6	**28.6**
Case6	203k	1313.2	739.6	100.5	**42.8**
Case7	232k	32.6	15.9	4.4	4.6
Case8	123k	65.2	45.8	19.4	**14.3**
Case9	202k	22.6	15.7	40.2	**4.5**
Case10	224k	29.4	12.8	7.5	**4.2**
Average Time	/	477.2	208.3	54.2	**25.0**

Our new approach:

- Compare to TG-Pro: reduce 94.7%
- Compare to incremental TG-Pro: reduce 87.9%
- Compare to the commercial tool, reduce 53.7%.

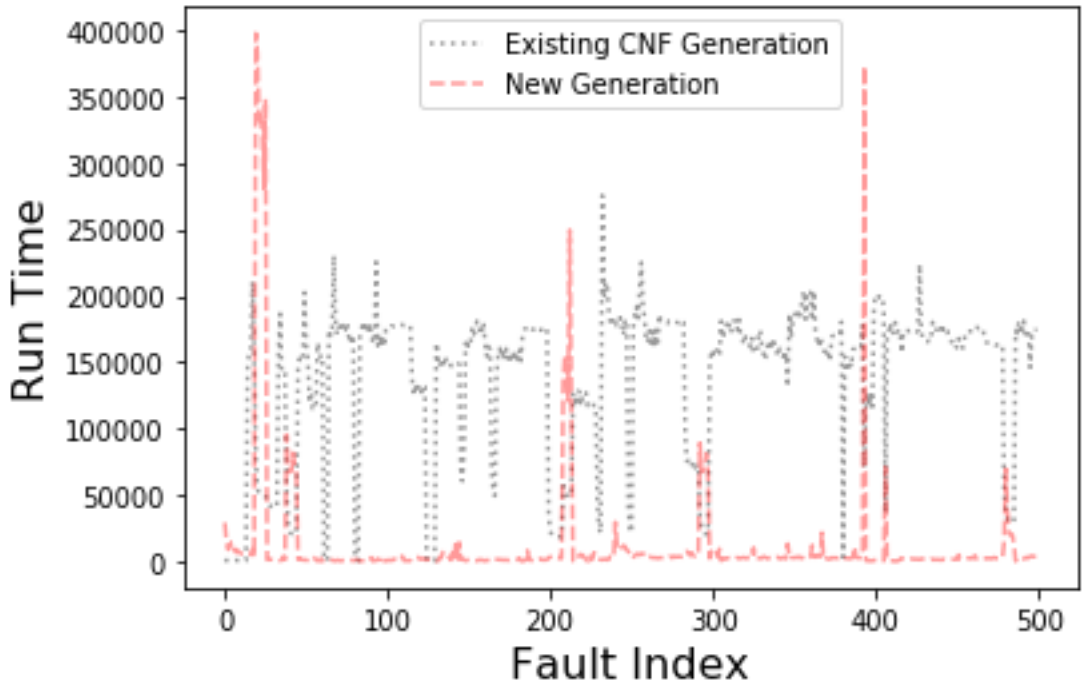
Experimental Results

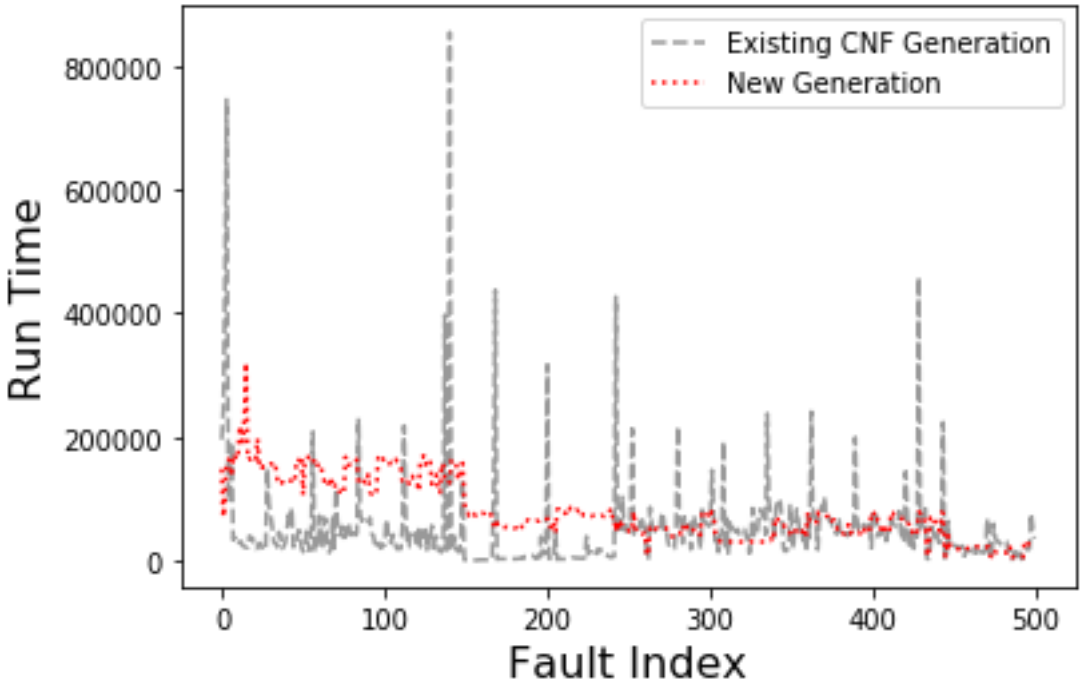
TABLE II
EVALUATION ON ATPG EFFECTIVENESS

Circuit	#gates	New Approach	Commercial	Improved Rate
Case1	4891k	100%	0.13%	99.87%
Case2	221k	100%	0.26%	99.74%
Case3	206k	100%	0%	100%
Case4	246k	100%	9.17%	90.83%
Case5	785k	100%	0.18%	99.82%
Case6	203k	100%	33.92%	66.08%
Case7	232k	100%	13.11%	86.89%
Case8	123k	100%	6.82%	93.18%
Case9	202k	100%	22.42%	77.58%
Case10	224k	100%	17.26%	82.74%
Average Eff	/	100%	10.33%	89.67%

- Our new approach can reach full coverage.
- Compare to commercial tool: improve the successful rate of solving hard-to-detect faults by 89.67%.

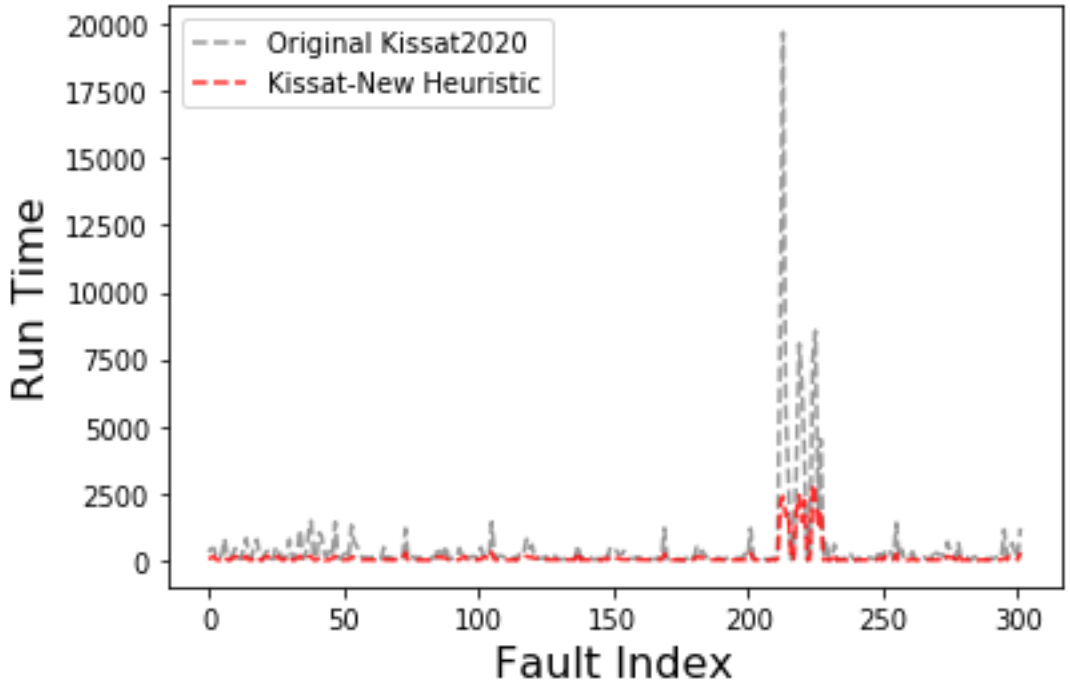
Experimental Results

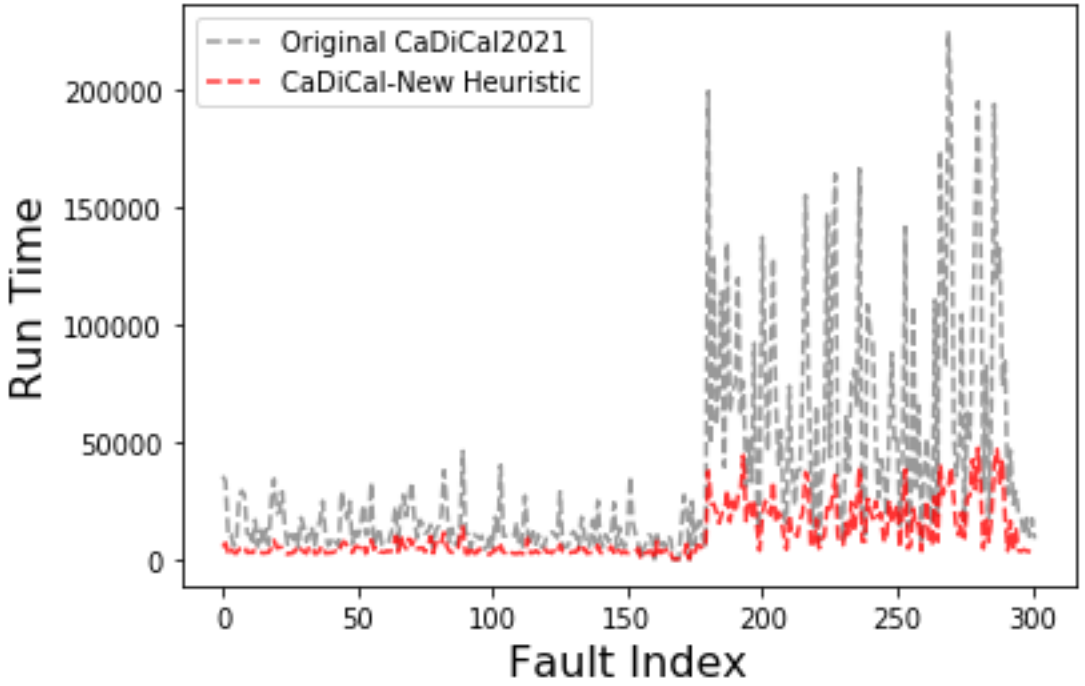




Circuit-to-CNF generation time for redundant faults (left) and aborted faults (right) can be reduced.

Experimental Results

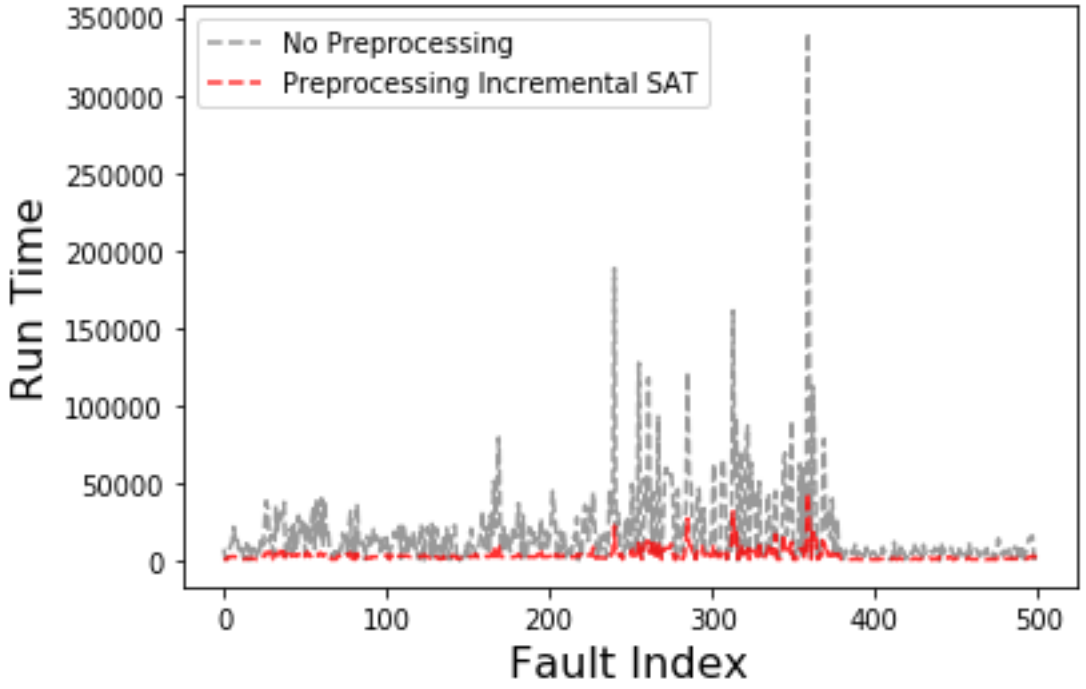


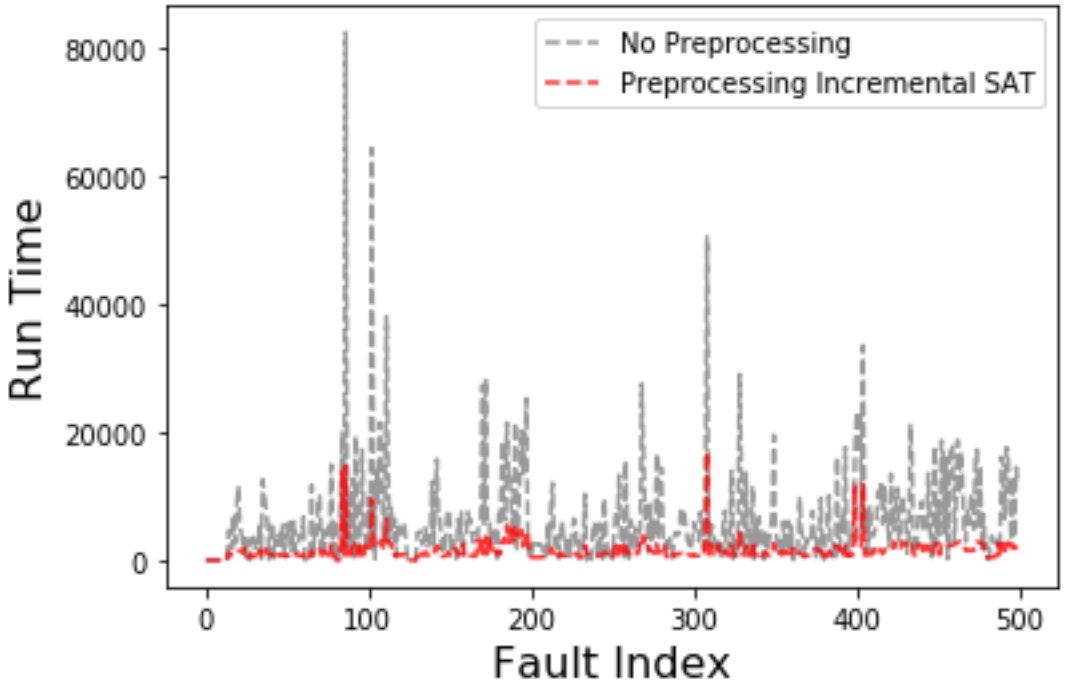


New CNF solving time for redundant faults (left) and aborted faults (right) can also be reduced.

Our method can be merged into any SAT solver.

Experimental Results





Preprocessing for redundant faults (left) and aborted faults (right) can also be helpful.

Outline

- Introduction of SAT-based ATPG
- Research Background on SAT-based ATPG
- Our New Framework
- Experimental results
- Conclusions

Conclusions

- We can have more efficient SAT-based ATPG frameworks.
  - Circuit-to-CNF generation is the real bottleneck of industrial applications of SAT-based ATPG
  - make good use of some methods from logic synthesis, for CNF reduction.
  - Joint optimization from the circuit structure to CNF solving can help.
  - Solving methods like cube-and-conquer can work better by considering the structure

Accelerate SAT-based ATPG via Preprocessing and New Conflict Management Heuristics

Junhua Huang Hui-Ling Zhen Naixing Wang

Mingxuan Yuan Hui Mao Yu Huang Jiping Tao