Trace Compaction using SAT-based Reachability Analysis

Sean Safarpour, Andreas Veneris, Hratch Mangassarian

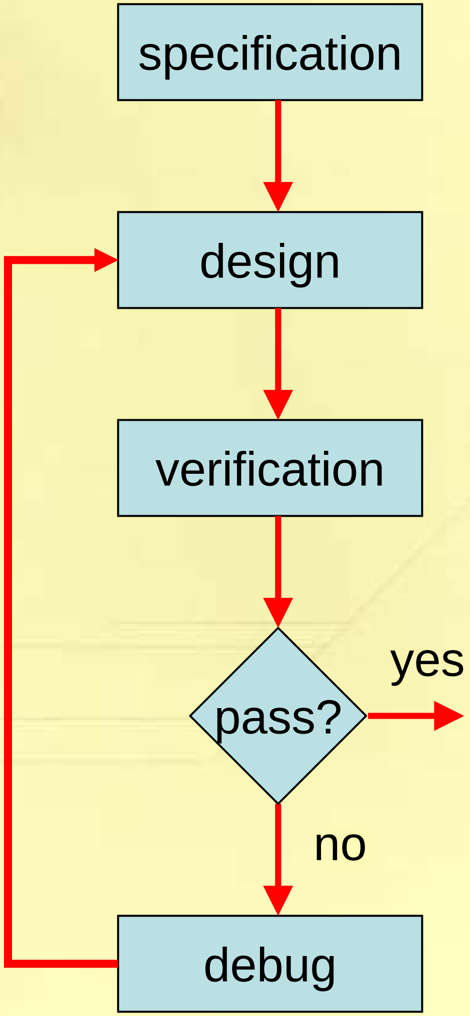
University of Toronto

Outline

- Introduction
- SAT-based Reachability Analysis
- Storing Visited States
- Experiments
- Conclusion

Introduction

- Verification determines if design is correct or not
  - simulation, formal and hybrid techniques exist
  - simulation is most popular
- If verification fails, must determine error source
  - called debugging
  - mostly manual
  - very time consuming



Introduction

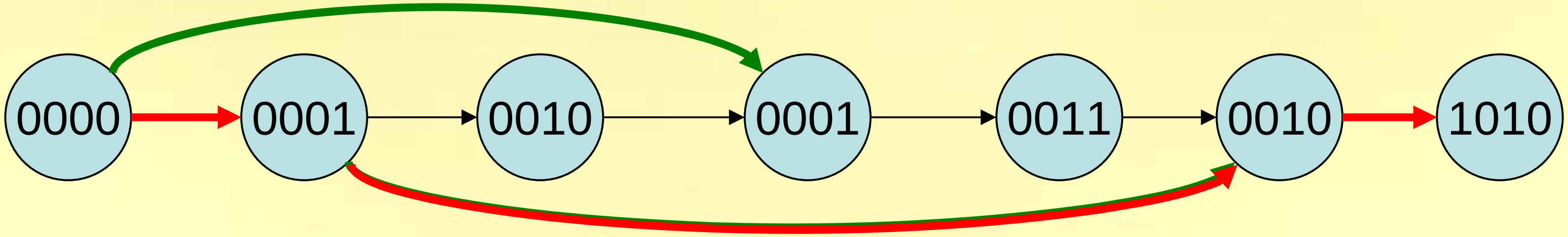
- Debugging is performed by
  - comparing expected and actual response
  - tracing signals and values
  - analyzing circuit behavior over all clock cycles
- A *trace* is the circuit behavior under a set of input stimulus
  - Output of verification tool: simulation/error trace
  - length of trace: number of clock cycles

Introduction

- Simulation Trace
  - starts from an initial state q_0
  - ends at final state q_k (where error is observed)
  - often generated from (semi) random simulation
- Long traces are harder to debug
  - more signals and values to analyze
- Trace Compaction/Reduction
  - reduce number of clock cycles required to observe error
  - increase debugging efficiency: save time and money

Introduction

- Trace compaction
  - determine if there exist a shorter path from initial state to final state
  - simple approach:
    - identify repeated states
    - add new transitions to repeated state
    - apply shortest path algorithm



SAT-based Reachability Analysis

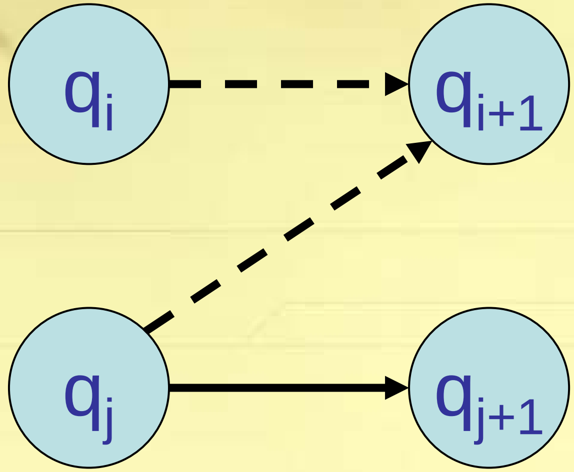
- Reachability Analysis:
  - can state q_i be reached from initial state q_0?
  - *pre-image* computation is central engine:
    - determine states that reach q_i in 1 transition
    - formulate problem as all-solution SAT
    - use observability don't cares for small solutions cubes
  - state space can be traversed in many manners
    - depth first, breadth first, heuristics-based traversal
    - traversal is key to efficiency of approach

SAT-based Reachability Analysis

- Reachability analysis for trace compaction
  - represent trace as graph G(N,E) with states N and transition E
  - find new states using pre-image computation
  - add new states and transitions to G
  - find shortest path from q_0 to q_k in G
- Follow rules to add new states to G
  - 3 rules depending on state covering relation

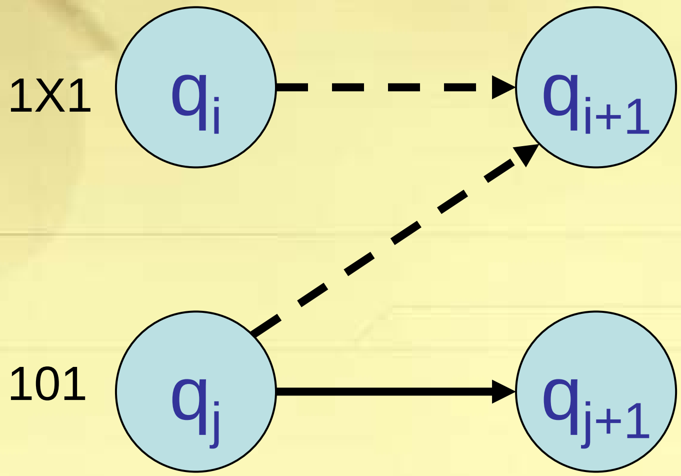
SAT-based Reachability Analysis

- Rule 1: new state q_i = previous state q_j
  - q_i is not added to G, but edge added from $q_j \rightarrow q_{i+1}$



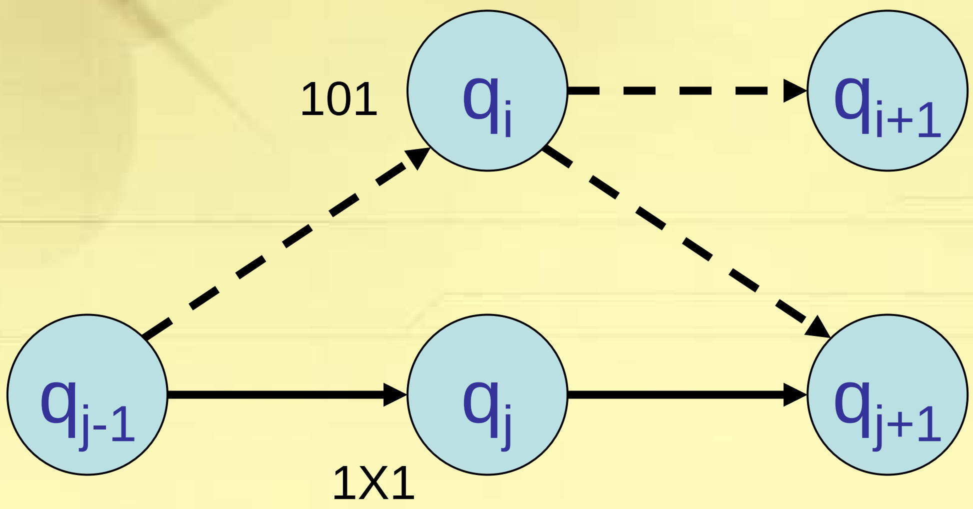
SAT-based Reachability Analysis

- Rule 2: new state q_i > previous state q_j
  - q_i is added to G, edge added from $q_i \rightarrow q_{i+1}$ and from $q_j \rightarrow q_{i+1}$



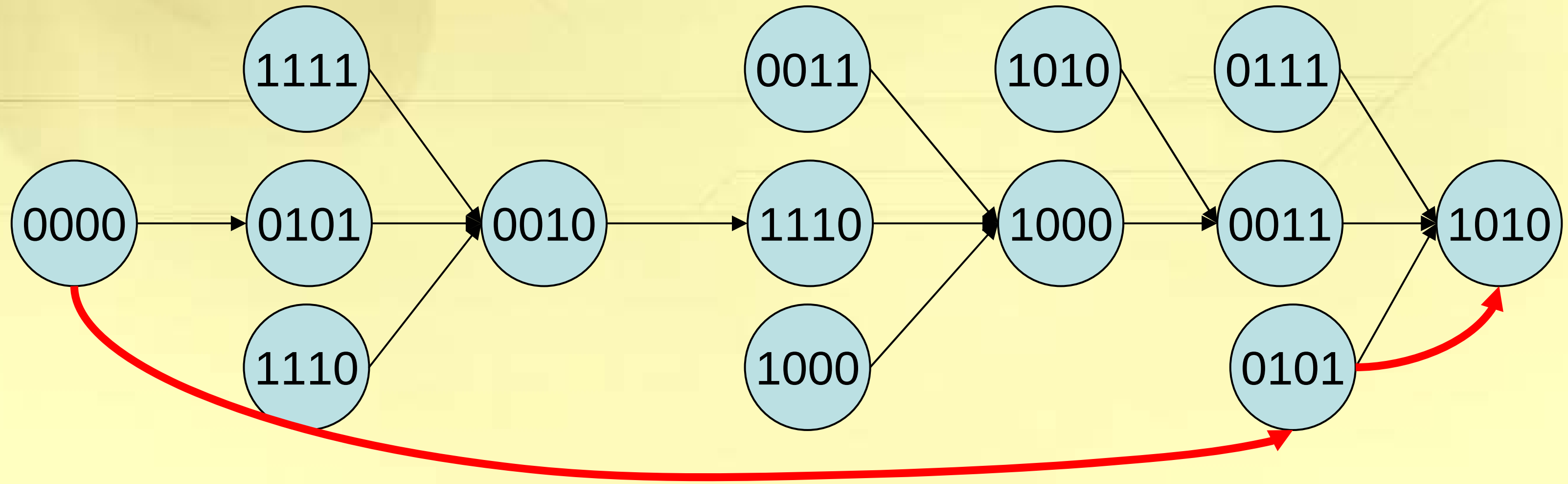
SAT-based Reachability Analysis

- Rule 3: new state q_i < previous state q_j
  - q_i is added to G, edge added from $q_i \rightarrow q_{i+1}$ and from $q_{j-1} \rightarrow q_i$ from $q_i \rightarrow q_{j+1}$



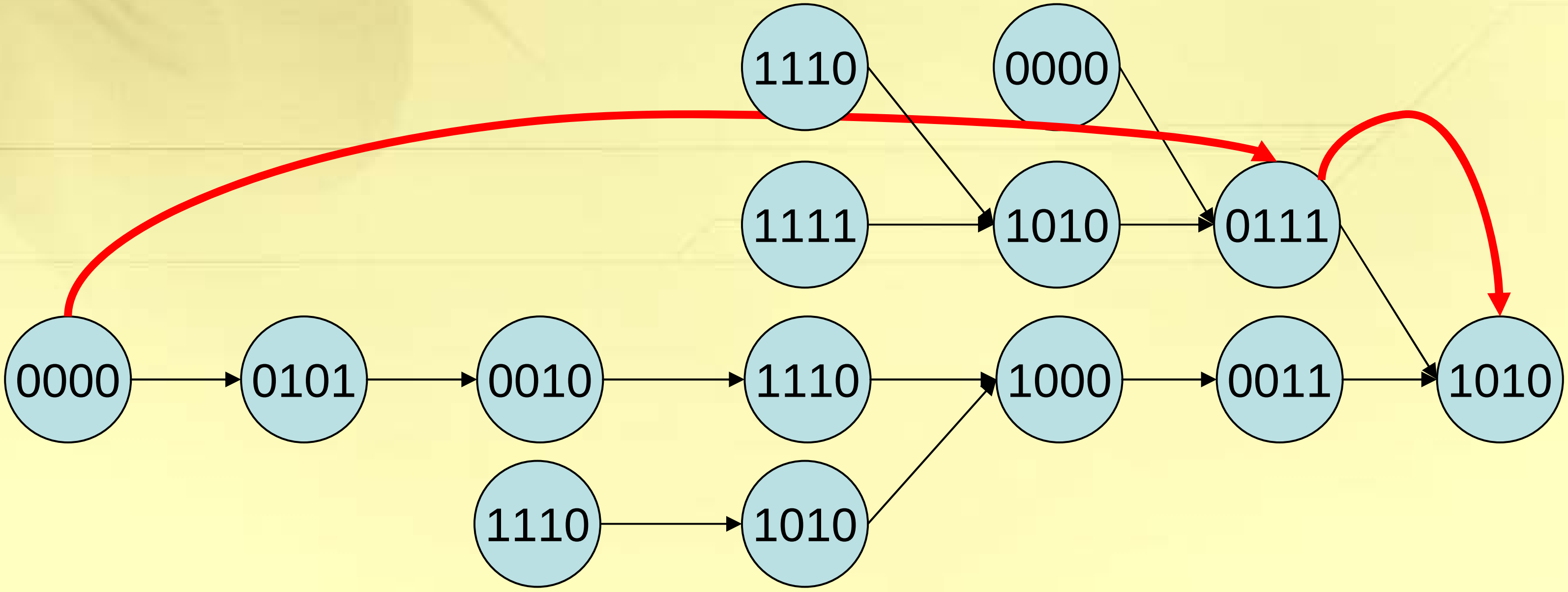
SAT-based Reachability Analysis

- The more states exist in the trace the higher the likelihood of adding new edges in G
- Increase likelihood by populating G
  - perform single pre-image step for all states



SAT-based Reachability Analysis

- Apply reachability analysis on the new G
- Goal is to find q_0: pick state with smallest hamming distance to q_0



Storing Visited States

- Need *containment relationships* between new found states q_i and old states q_j
  - $q_i = q_j$ (case 1), $q_i > q_j$ (case 2), $q_i < q_j$ (case 3)
- For a given state, need to find
  - what states cover it
  - what states are covered by it
- Must be space and time efficient

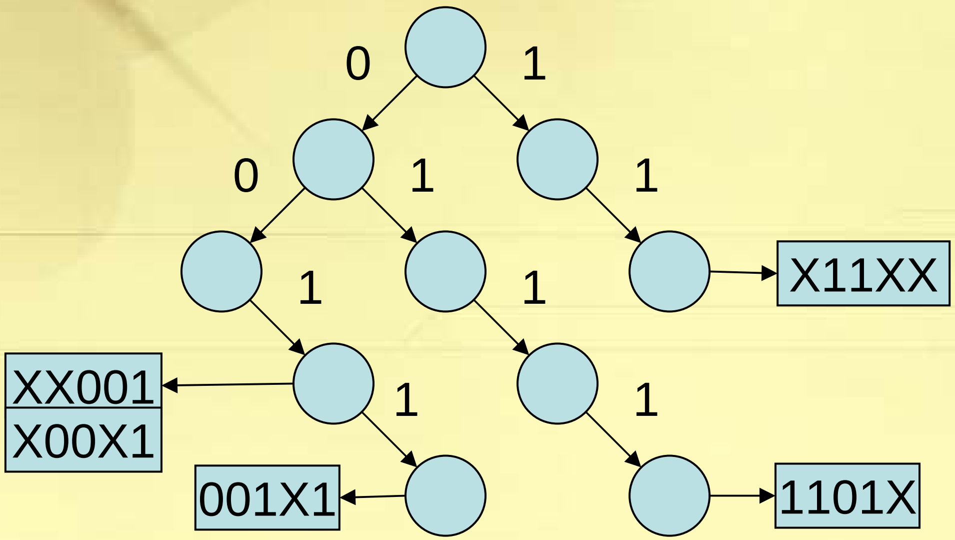
Storing Visited States

- Tree & hash table data structure proposed
  - Order states: 0 > 1 > X (>: more significant than)
  - Call these *ordered cubes*
- Store states in tree based on ordered cube's # 0's, # 1's, # X's
  - Left edge represents 0
  - Right edge represent 1
  - No edges for X

Storing Visited States

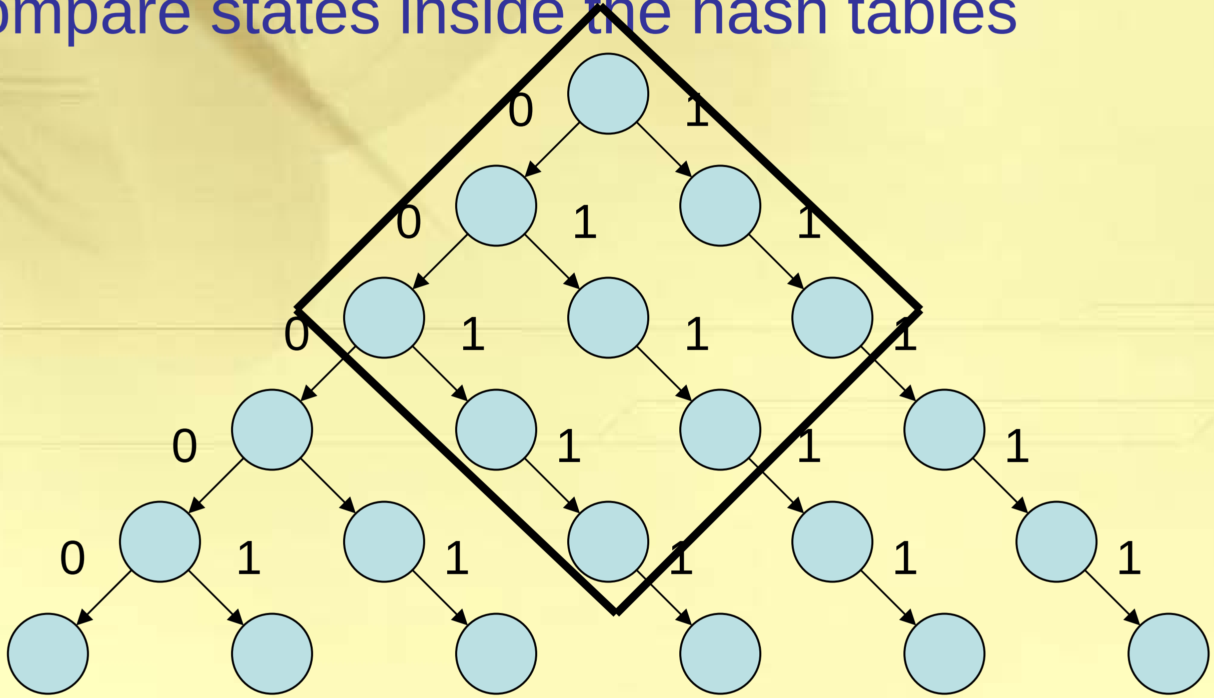
- Example:

states	1101X	001X1	XX001	X00X1	X11XX
ordered cube	0111X	0011X	001XX	001XX	11XXX



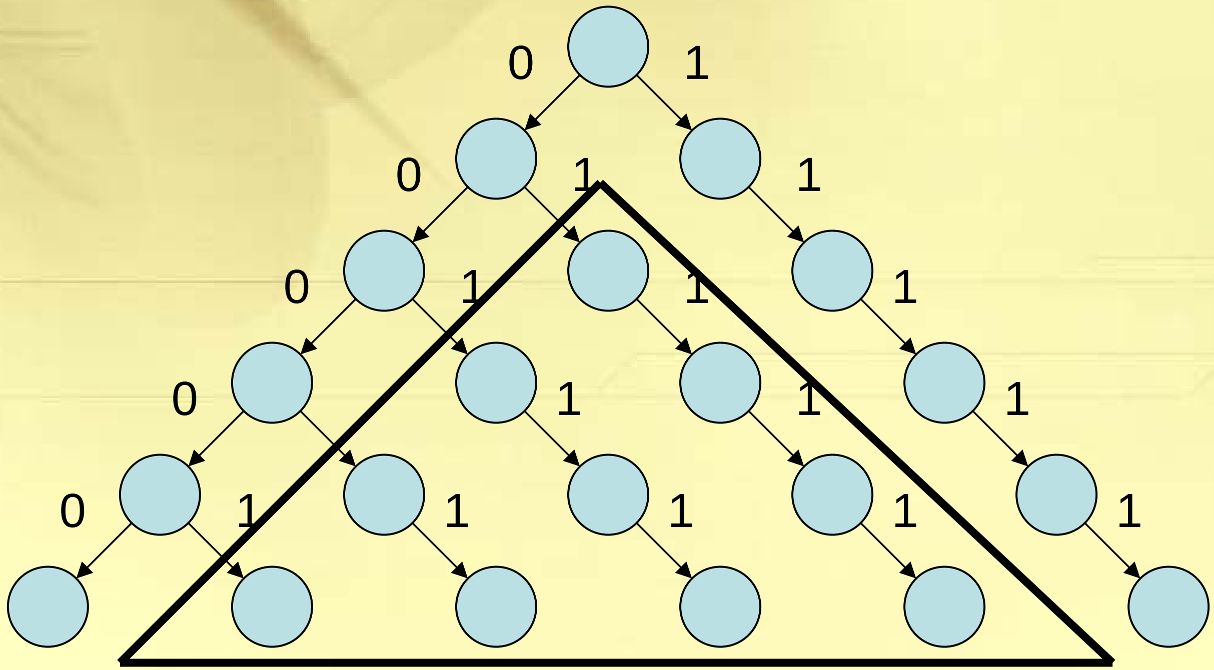
Storing Visited States

- Finding supersets
  - nodes with at least as many X’s
  - rectangle with area ≈ # 1’s x # 0’s
  - compare states inside the hash tables



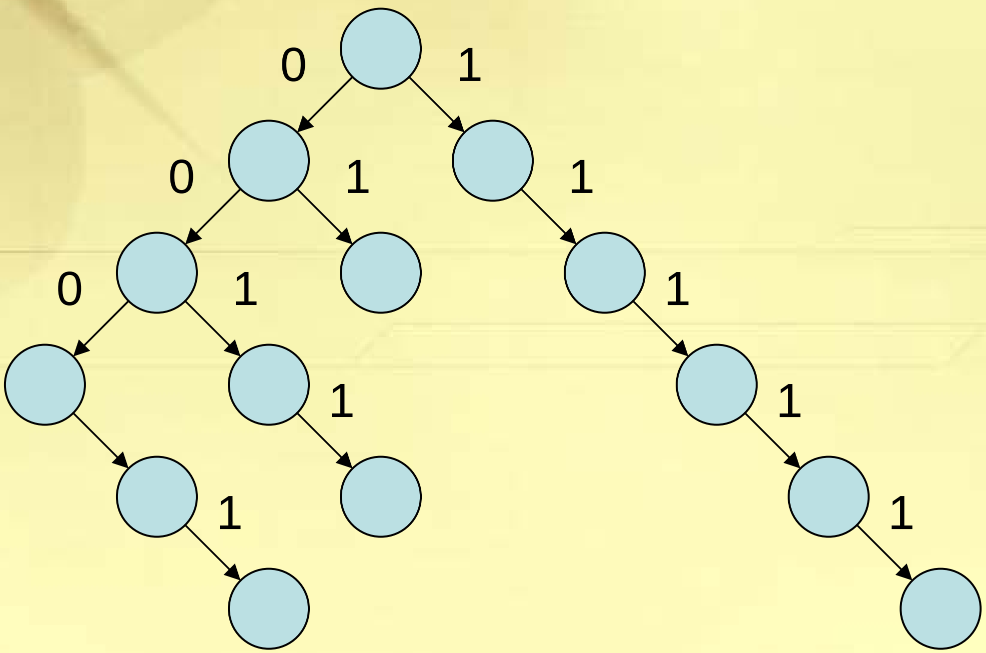
Storing Visited States

- Finding subsets
  - nodes with at most as many X's
  - triangle with area ≈ (# X's)^2 /2
  - compare states inside the hash tables



Storing Visited States

- Tree is usually not fully populated
- Comparison with states in hash table is fast
- Data structure effective in practice

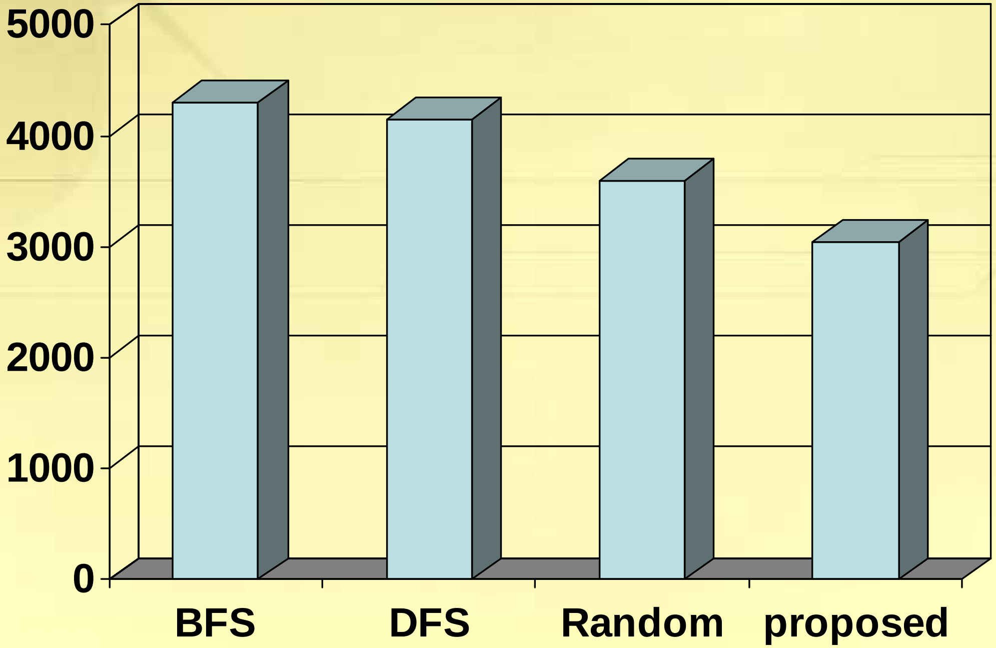


Experiments

- Traces of length 50,100,1000 from random simulation
- circuit from ISCAS'89 and ITC'99
- all-solution SAT solvers uses *Observability Don't Cares* to reduce state cubes [Safarpour DATE'04]
- Limit of 10,000 state found used instead of explicit memout or timeout

Experiments

- Evaluate state selection heuristic:
  - smallest hamming distance
- Sum of run time of reachability analysis problems with timeout 200 sec for each problem



Experiments

- Result summary for trace reduction problems

	original length 50			original length 100			original length 1000		
	avg red	affected	red	avg red	affected	red	avg red	affected	red
pre	10.1	70%	13.8	16.9	72%	22.7	266.3	71%	362.8
reach	3.8	37%	8.5	6.1	35%	15.4	2.8	15%	12.4
both	19.7	74%	25.7	36.2	72%	49.0	327.8	72%	446.6

Conclusion

- Pre-image computation & reachability analysis can help trace reduction
  - use all-solution SAT solver, good state selection heuristic
- Must determine how to benefit from new found state:
  - develop 3 rules
- Must determine containment relationship efficiently:
  - develop tree and hash table data structure
- Experiments confirm effectiveness of approach