Provably Optimal Test Cube Generation using Quantified Boolean Formula Solving

ASP-DAC 2013

Albert-Ludwigs-Universität Freiburg

Matthias Sauer, Sven Reimer, Ilia Polian, Tobias Schubert, Bernd Becker

Chair of Computer Architecture

01.24.13

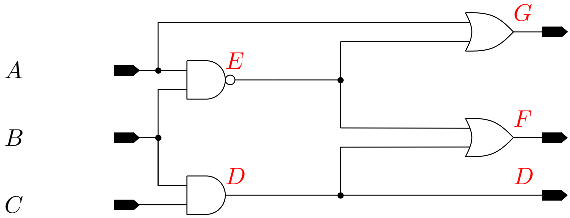
Motivation – Test pattern relaxation

- Test cube:
  - Parts of the pattern are unspecified (Don't Care)
  - Test requirements still hold
- Used for:
  - Refilling
  - Minimizing power consumption
  - Compaction (e.g., Embedded Deterministic Test)
- All known techniques are approximative
- Our approach:
  - Test cube generation with maximum number of Don't Cares

  $\Rightarrow$ Optimal test cube
  - Measure the quality of heuristic methods

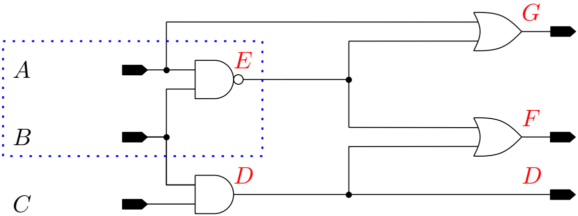
Outline

Circuit encoding



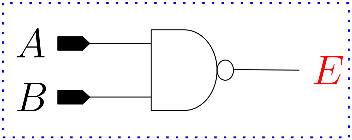
- Boolean satisfiability (SAT) formulation in CNF: Tseitin encoding [Tseitin ’68]
- Additional variables for each gate
- Linear in circuit size

Circuit encoding



- Boolean satisfiability (SAT) formulation in CNF: Tseitin encoding [Tseitin '68]
- Additional variables for each gate
- Linear in circuit size

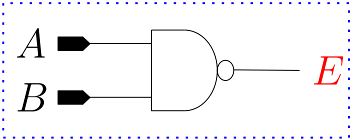
Circuit encoding



- Boolean satisfiability (SAT) formulation in CNF: Tseitin encoding [Tseitin ’68]
- Additional variables for each gate
- Linear in circuit size

Circuit encoding

$$E \leftrightarrow \neg(A \wedge B)$$

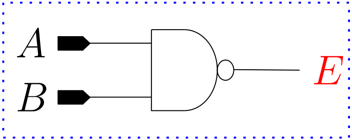


- Boolean satisfiability (SAT) formulation in CNF:
  Tseitin encoding [Tseitin '68]
- Additional variables for each gate
- Linear in circuit size

Circuit encoding

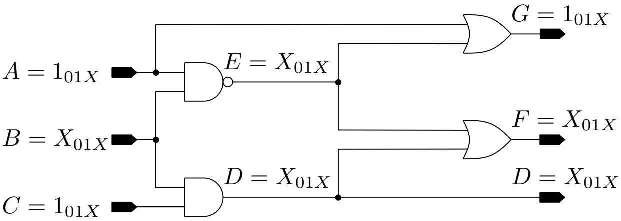
$$(A \vee E) \wedge \overbrace{(B \vee E)}^{Clause} \wedge (\overbrace{\neg A}^{Literal} \vee \neg B \vee \neg E)$$

$$E \leftrightarrow \neg(A \wedge B)$$



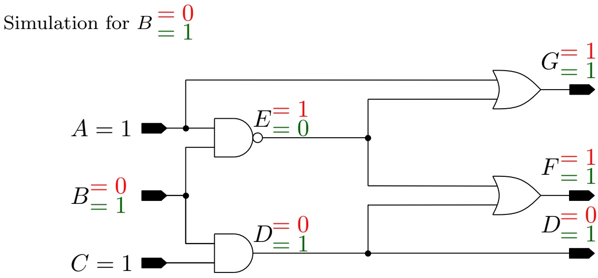
- Boolean satisfiability (SAT) formulation in CNF: Tseitin encoding [Tseitin '68]
- Additional variables for each gate
- Linear in circuit size

Unspecified values – $01X$ logic [Jain et al. '00]



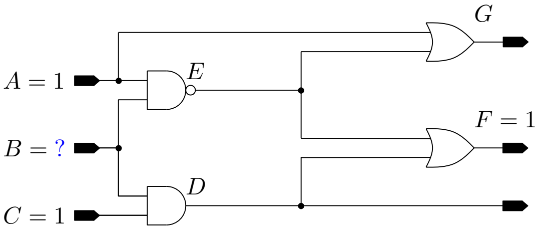
- Three-valued logic:
  0_{01X} (logic 0), 1_{01X} (logic 1), X_{01X} (unknown)
- $01X$ in SAT: $0_{01X} = (0,1)$, $1_{01X} = (1,0)$, $X_{01X} = (0,0)$
- SAT encoding for $01X$ doubles size of the formula
- In example: Output F is unknown if input B is unspecified

Unspecified values – Exact formulation



- But: *F* can be set to 1, even if *B* is unspecified

Unspecified values – Exact formulation

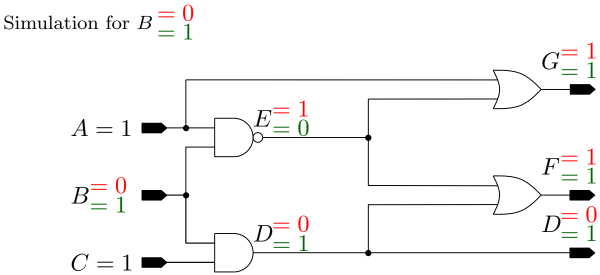


- But: *F* can be set to 1, even if *B* is unspecified

⇒ QBF: Universally quantified variables for unknown values

- $\underbrace{\exists\{A,C\}\forall\{B\}\exists\{D,E,F,G\}}_{\text{Prefix}}.\ \underbrace{\varphi(A,\ldots,G)}_{\text{Tseitin encoding}} \wedge (A) \wedge (C) \wedge \underbrace{(F)}_{\text{property}}$

Unspecified values – Exact formulation



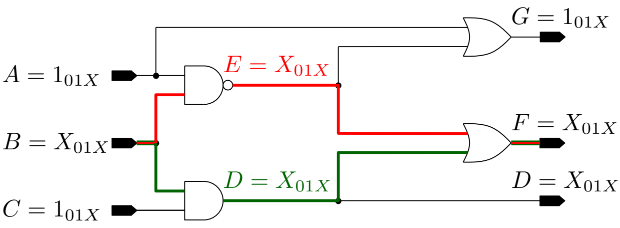
- But: F can be set to 1, even if B is unspecified

⇒ QBF: Universally quantified variables for unknown values

- $\underbrace{\exists\{A,C\}\forall\{B\}\exists\{D,E,F,G\}}_{\text{Prefix}}.\ \underbrace{\varphi(A,\ldots,G)}_{\text{Tseitin encoding}} \wedge (A) \wedge (C) \wedge \underbrace{(F)}_{\text{property}}$

- QBF: reconvergent paths are resolved by formulation

Unspecified values – Exact formulation



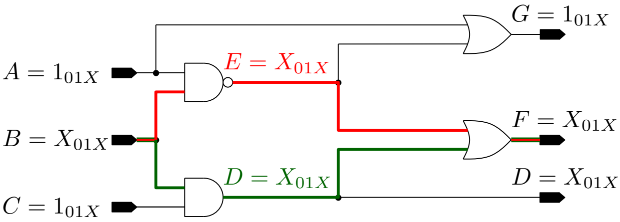
- But: F can be set to 1, even if B is unspecified

⇒ QBF: Universally quantified variables for unknown values

- $\underbrace{\exists\{A, C\}\forall\{B\}\exists\{D, E, F, G\}}_{\text{Prefix}} . \underbrace{\varphi(A, \ldots, G)}_{\text{Tseitin encoding}} \wedge (A) \wedge (C) \wedge \underbrace{(F)}_{\text{property}}$

- QBF: reconvergent paths are resolved by formulation
  01X: reconvergent paths may block propagation of values

Unspecified values – Exact formulation



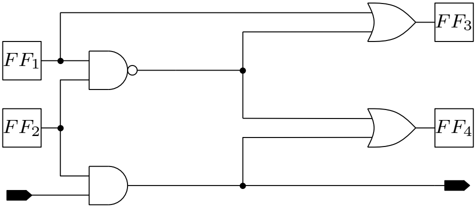
- But: F can be set to 1, even if B is unspecified

⇒ QBF: Universally quantified variables for unknown values

- $\underbrace{\exists\{A, C\}\forall\{B\}\exists\{D, E, F, G\}}_{\text{Prefix}} . \underbrace{\varphi(A, \ldots, G)}_{\text{Tseitin encoding}} \wedge (A) \wedge (C) \wedge \underbrace{(F)}_{\text{property}}$

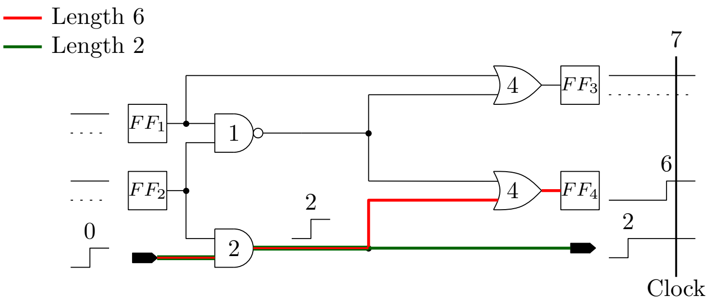
- QBF: Exact formulation for Don't Cares
  $01X$: Approximative formulation for Don't Cares

Sensitizable paths + small delay faults



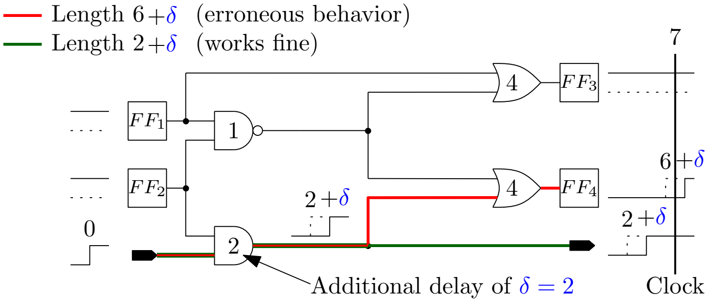
- Sensitizable path: Transition from input to output
- Length of a path according to sum of gate delays

Sensitizable paths + small delay faults



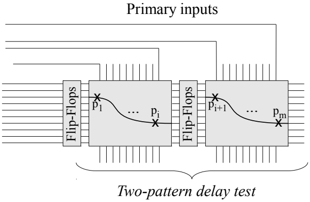
- Sensitizable path: Transition from input to output
- Length of a path according to sum of gate delays

Sensitizable paths + small delay faults



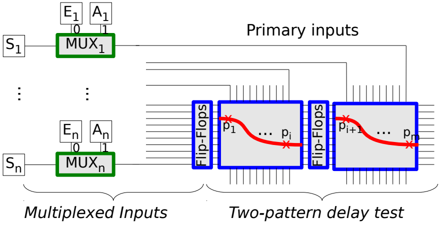
- Sensitizable path: Transition from input to output
- Length of a path according to sum of gate delays
- Small delay faults: Assume additional delay for one gate
- Output transition too late for clock
- Two-pattern delay test
- The longer the path the higher the detection quality

Optimal test cube generation



- Small delay faults over two timeframes
- Test cube with maximum number of unspecified inputs using QBF
- Quantify unspecified inputs universally, specified ones existentially
- If path for small delay fault is sensitizable:
  Universally quantified inputs: excluded from test cube
  Existential quantified inputs: test cube
- But: The quantifier of a variable cannot be changed in QBF
  Unspecified inputs are unidentified a-priori
  Which inputs have to be quantified universally?

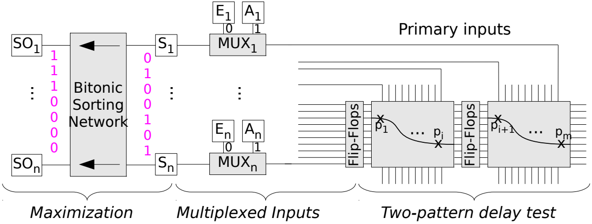
Multiplexed inputs



$$\psi = \exists\{S_1,\ldots,S_n,E_1,\ldots,E_n\}\forall\{A_1,\ldots,A_n\}\exists\ldots\varphi_{Circuit} \wedge \varphi_{Property} \wedge \varphi_{MUX}$$

- Dynamic choice of (un-)specified input with multiplexer
- Select input S_i switches between specified ($S_i = 0 : \exists E_i$) and unspecified ($S_i = 1 : \forall A_i$) for any primary input I_i
- Find the maximum number of select inputs that can be set to 1

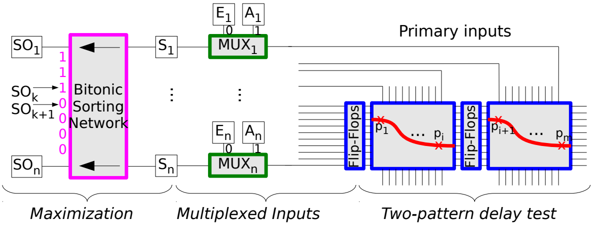
Maximization



- Sort select-inputs S_i with Bitonic sorting network [Batcher '68]
- Circuit size of sorter: $O(n \log n)$
- Input vector $\overrightarrow{S}$ is sorted by 1's and 0's

⇒ Sorted output vector $\overrightarrow{SO}$

Optimal test cube generation

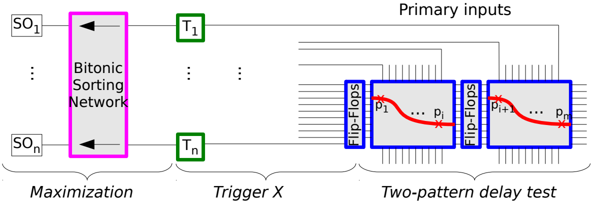


$$\psi(j) \quad = \quad \exists\{SO_1,\ldots,SO_n,S_1,\ldots,S_n,E_1,\ldots,E_n\}\forall\{A_1,\ldots,A_n\}\exists\ldots$$
$$\varphi_{Circuit} \wedge \varphi_{Property} \wedge \varphi_{MUX} \wedge \varphi_{Sorter} \wedge (SO_j)$$

$\Rightarrow$ Binary search over j

- Search for k, such that: path is sensitizable with k unspecified inputs ($SO_k = 1$), but not with $k+1$ ($SO_{k+1} = 0$)
- QBF solver returns assignment for outermost existential variables: $S_1,\ldots,S_n$: unspecified inputs; remaining $E_1,\ldots,E_n$: test cube
- Optimal test cube, i.e., maximum number of Don't Cares

$01X$-Optimal test cube generation



$$\varphi(j) \quad = \quad \underbrace{\varphi_{Circuit}}_{01X \text{ encoding}} \wedge \varphi_{Property} \wedge \varphi_{Trigger} \wedge \varphi_{Sorter} \wedge (SO_j)$$

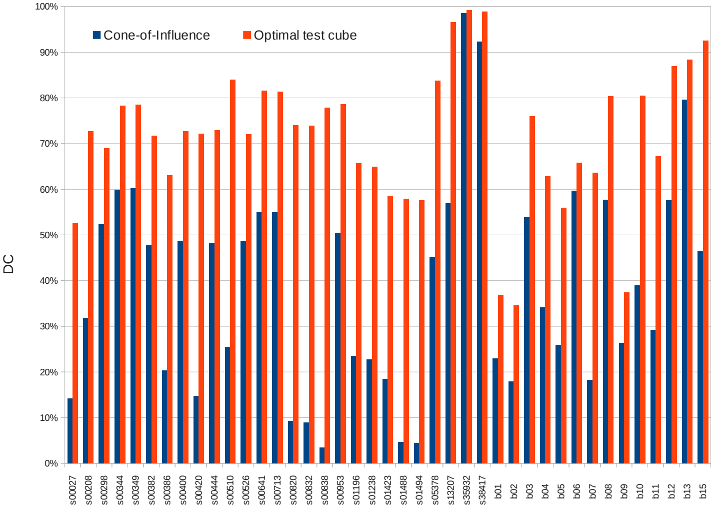
⇒ Binary search over j

- Search for k, such that: path is sensitizable with k unspecified inputs ($SO_k = 1$), but not with $k+1$ ($SO_{k+1} = 0$)
- If $T_i = 1$, corresponding input I_i is set to X_{01X}
- SAT solver returns assignment for all variables: $T_1, \ldots, T_n$: unspecified inputs; remaining input variables: test cube
- $01X$-Optimal test cube, i.e., optimal for $01X$ encoding

Experimental setup

- Sequential versions of ISCAS 89 and ITC 99 benchmarks
- SAT-based path generator `PHAETON` [Sauer et al. '11]:
  100 longest broadside testable paths of each circuit
- In-house SAT solver `antom` [Schubert et al. '10] and
  QBF solver `quantom`
- Cone-of-influence (COI) reduction
- Average percentage of Don't Cares (DC)

Results for ISCAS 89 & ITC 99 circuits



Comparison of QBF-optimal result

Static (initial test pattern needed):

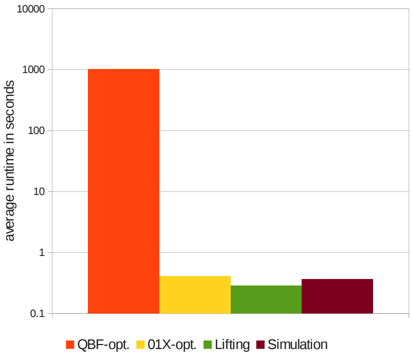
1. Lifting [Ravi, Somenzi ’04] (best case QBF-optimal)
2. Simulation (best case 01X-optimal)

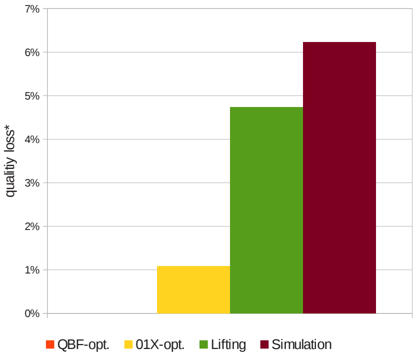
- Average over 100 random initial test patterns

Dynamic (find test cube directly with given test requirements):

3. 01X-optimal

Comparison





$$^{*}\mathit{Loss}(\text{Method}) = 1 - \frac{DC_{\text{Method}} - DC_{\text{COI}}}{DC_{\text{Optimal}} - DC_{\text{COI}}}$$

Conclusion

- Novel technique for generation test cubes with QBF
- First approach producing test cubes with maximum number of Don't Cares
- Framework adaptable to any task that maximizes number of unspecified lines
- Compare heuristic approaches with true optimum
- New and fast method for 01X encoding (01X-optimal)

Future work

- Adapt framework to other applications and fault models
- Increase scalability of QBF-solver

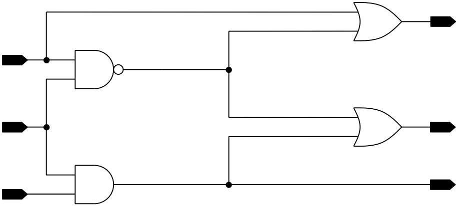
SAT + QBF

- Satisfiability problem or SAT problem:
  Given propositional formula φ. Is there an assignment to the variables, such that φ is satisfied?
- φ in conjunctive normal form (CNF), e.g.,
  $$\varphi(x_1, \ldots, x_n) = (x_1 \vee \underbrace{\neg x_2}_{\text{literal}}) \wedge \underbrace{(x_2 \vee x_3 \vee \neg x_4)}_{\text{clause}} \wedge \ldots$$
- Notation: $\varphi(x_1, \ldots, x_n) = \{\{x_1, \neg x_2\}, \{x_2, x_3, \neg x_4\}, \ldots\}$
- Properties of CNF:
  Clause is satisfied iff at least one literal is assigned to 1.
  CNF is satisfied iff all clauses are satisfied.
- Combinational circuits can be transformed into CNF in linear size of the circuit (Tseitin encoding)
- Well known NP-complete problem with enormous improvements in the last decades

SAT + QBF

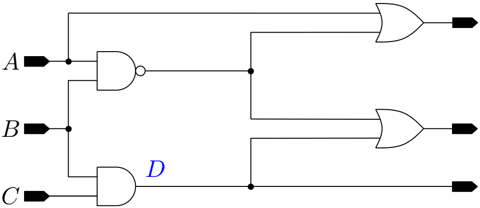
- Quantified Boolean formula (QBF) is an extension of SAT: variables are quantified existentially ($\exists$) or universally ($\forall$)
- Example for a QBF ψ in prenex normal form:
$$\psi(x_1,\ldots,x_n) = \underbrace{\exists\{x_1\}\forall\{x_2,x_3\}\exists\{x_4\}\ldots\exists\{x_n\}}_{\text{prefix}}.\underbrace{\varphi(x_1,\ldots,x_n)}_{\text{matrix (CNF)}}$$
- Semantics (for this example):
ψ is satisfied iff there exists one assignment for x_1 such that for every assignment of x_2 and x_3, there exists one assignment for x_4 and so forth, such that φ is satisfied.
- PSPACE-complete problem with increasing interest in the last decade

Circuit encoding



- Circuit to propositional formulae in CNF via Tseitin encoding [Tseitin '68]

Circuit encoding

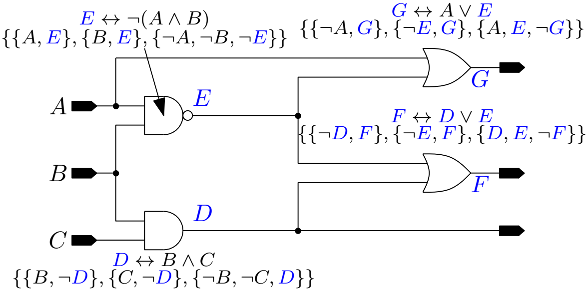


$D \leftrightarrow B \wedge C$

$\{\{B, \neg D\}, \{C, \neg D\}, \{\neg B, \neg C, D\}\}$

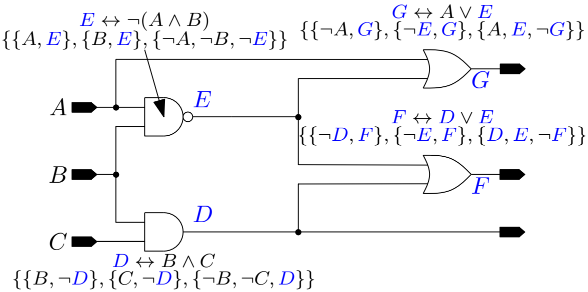
- Circuit to propositional formulae in CNF via Tseitin encoding [Tseitin '68]
- Introduces additional Tseitin variables

Circuit encoding



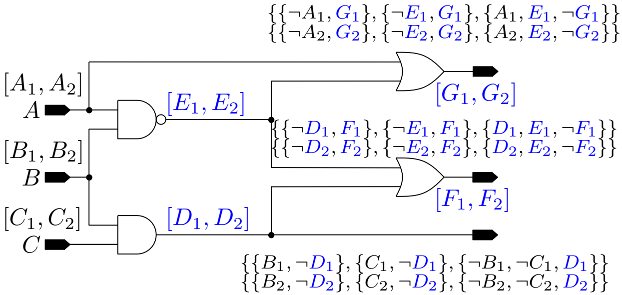
- Circuit to propositional formulae in CNF via Tseitin encoding [Tseitin '68]
- Introduces additional Tseitin variables
- Resulting formula is linear in circuit size

Encode small delay faults



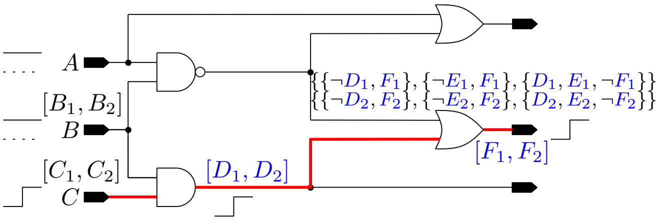
- Encode both timeframes …

Encode small delay faults



- Encode both timeframes …

Encode small delay faults



- Encode both timeframes …
- … and trigger path with unit clauses (in this example: $\{\{\neg C_1\}, \{C_2\}, \{\neg D_1\}, \{D_2\}, \{\neg F_1\}, \{F_2\}\}$)

Literature

[Batcher '68] K. E. Batcher, "Sorting networks and their applications," in AFIPS Spring Joint Computing Conference, pp. 307–314, ACM, 1968.

[Jain et al. '00] A. Jain, V. Boppana, R. Mukherjee, J. Jain, M. Fujita, and M. S. Hsiao, "Testing, Verification, and Diagnosis in the Presence of Unknowns," in VLSI Test Symp., pp. 263–269, 2000.

[Ravi, Somenzi '04] K. Ravi and F. Somenzi, "Minimal assignments for bounded model checking," in Tools and Algorithms for the Construction and Analysis of Systems, vol. 2988, pp. 31–45, Springer, 2004.

[Sauer et al. '11] M. Sauer, A. Czutro, T. Schubert, S. Hillebrecht, I. Polian, and B. Becker, "SAT-based analysis of sensitisable paths," in IEEE Design and Diagnostics of Electronic Circuits and Systems, pp. 93–98, 2011.

[Schubert et al. '10] T. Schubert, M. Lewis, and B. Becker, "antom — Solver Description," in SAT Race, 2010.

[Tseitin '68] G.S. Tseitin, "On the complexity of derivations in propositional calculus," in Studies in Constructive Mathematics and Mathematical Logics, 1968.