Generating High-Quality Tests for Boolean Circuits by Treating Tests as Proof Encoding

Eugene Goldberg, Pete Manolios
Northeastern University, USA

An extended version of the TAP-2010 talk

References

- E.Goldberg, P.Manolios. *Generating High-Quality Tests for Boolean Circuits by Treating Tests as Proof Encoding,* TAP-2010, Malaga, Spain, LNCS 6143, pp.101-116

- E.Goldberg. *On bridging simulation and formal verification,* VMCAI-2008, San Francisco, USA, LNCS 4905, pp.127-141

- E.Goldberg. *Testing Satisfiability of CNF Formulas by Computing a Stable Set of Points.* Proceedigns of Conference on Automated Deduction, CADE 2002,pp.161-180 .

Summary

- Introduction
- Test generation algorithm based on TTPE
- Experimental results and conclusion

Motivation

- Testing is easy (scalable) but incomplete
- Formal verification is complete but hard
- How do we get the best of both worlds (e.g. by generating a small test set that is complete or close to such)?

- We study testing by the example of combinational circuits
- We describe circuits by propositional logic

Main Idea

TTPE: Treating Tests As Proof Encoding

Q: Why does testing work at all?

A: Short proof $\Rightarrow$ small encoding test set

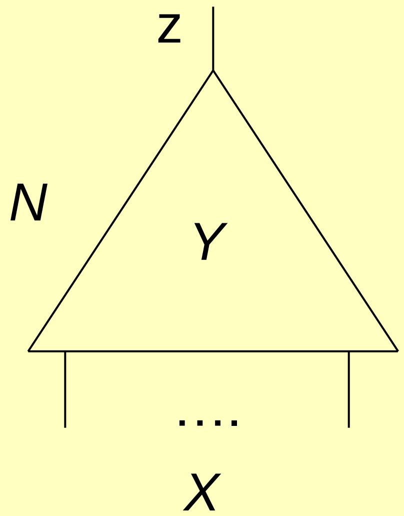
Q: What is the best coverage metric?

A: A formal proof is an ideal coverage metric (contains a complete set of corner cases)

Q: When does one stop generating tests?

A: When the test set encodes a proof

Description of the Setup



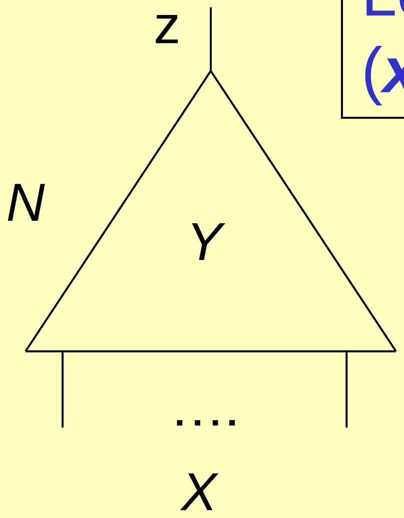
N is a combinational circuit,
It is correct if $N \equiv 0$,
It has a bug if $N(\boldsymbol{x}) = 1$.

Circuit $N \Rightarrow$ CNF formula $F^N(X,Y,z)$,

$N \equiv 0 \Leftrightarrow (F^N \wedge z) \equiv 0$

Point $\boldsymbol{p}$ is $(\boldsymbol{x},\boldsymbol{y},z)$ i.e. a complete assignment to $X \cup Y \cup \{z\}$
The test extracted from $\boldsymbol{p}$ is $\boldsymbol{x}$ (i.e. assignment to X)

A Naive Testing Procedure



Let $F = F_N \wedge z$. Given test $\boldsymbol{x}_i$, S_i is a set of points $(\boldsymbol{x}_i, \boldsymbol{y}, \boldsymbol{z})$ where the value of $\boldsymbol{x}_i$ is the same.

$N(\boldsymbol{x}_1)=0$, $Enc(F, S_1) =$ *no*

$N(\boldsymbol{x}_2)=0$, $Enc(F, S_1 \cup S_2) =$ *no*

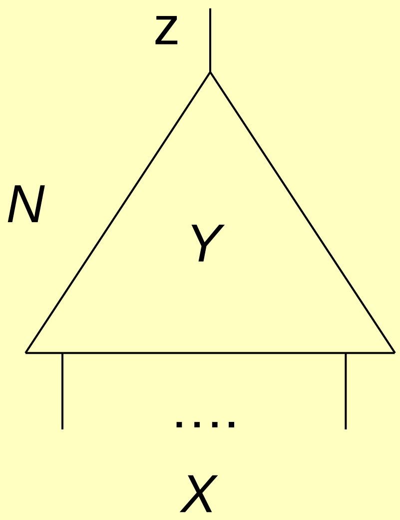
……..

$N(\boldsymbol{x}_i)=0$, $Enc(F, S_1 \cup .. \cup S_i) =$ *no*

$N(\boldsymbol{x}_{i+1})=1$, $Enc(F, S_1 \cup .. \cup S_{i+1}) =$ *no*

$N(\boldsymbol{x}_{i+1})=0$, $Enc(F, S_1 \cup .. \cup S_{i+1}) =$ *yes*

A Modified Version



$\boxed{i \neq j \Rightarrow \boldsymbol{p}_{\mathbf{i}} \neq \boldsymbol{p}_{\mathbf{j}} \text{ but } \boldsymbol{x}_{\mathrm{i}} \text{ may be equal to } \boldsymbol{x}_{\mathrm{j}}}$

$\boldsymbol{p}_{\mathbf{1}}, N(\boldsymbol{x}_1)=0, Enc(F,\{\boldsymbol{p}_{\mathbf{1}}\}) = \ no$

$\boldsymbol{p}_{\mathbf{2}}, N(\boldsymbol{x}_2)=0, Enc(F,\{\boldsymbol{p}_{\mathbf{1}},\boldsymbol{p}_{\mathbf{2}}\}) = \ no$

....

$\boldsymbol{p}_{\mathbf{i}}, N(\boldsymbol{x}_{\mathrm{i}})=0, Enc(F,\{\boldsymbol{p}_{\mathbf{1}},..,\boldsymbol{p}_{\mathrm{i}}\}) = \ no$

$\boldsymbol{p}_{\mathbf{i+1}}, N(\boldsymbol{x}_{\mathrm{i+1}})=1, Enc(F,\{\boldsymbol{p}_{\mathbf{1}},..,\boldsymbol{p}_{\mathrm{i+1}}\}) = \ no$

$\boldsymbol{p}_{\mathbf{i+1}}, N(\boldsymbol{x}_{\mathrm{i+1}})=0, Enc(F,\{\boldsymbol{p}_{\mathbf{1}},..,\boldsymbol{p}_{\mathrm{i+1}}\}) = \ yes$

Summary

- Introduction
- Test generation algorithm based on TTPE
- Experimental results and conclusion

High-Level Description

- We use the resolution proof system
- Checking if a set of points encodes a resolution proof is hard
- Instead, we generate points (called boundary) that encode mandatory fragments of a proof
- So, instead of trying to encode an entire proof we target essential parts of it
- We stop when a counterexample is found or a resource is exceeded

Encoding a Resolution Proof

$C' = y_1 \vee y_2 \vee \sim y_5$, $C'' = \sim y_1 \vee y_7 \quad \Rightarrow \quad C = y_2 \vee \sim y_5 \vee y_7$

Points $\boldsymbol{p}'$ and $\boldsymbol{p}''$ encode resolution *of* C' and C'' if:

$\boldsymbol{p}' = (y_1=0, y_2=0, y_5=1, y_7=0, \ldots\ldots.)$, $C'(\boldsymbol{p}')=0$
$\boldsymbol{p}'' = (y_1=1, y_2=0, y_5=1, y_7=0, \ldots\ldots.)$, $C''(\boldsymbol{p}'')=0$
$Hamming\_distance(\boldsymbol{p}', \boldsymbol{p}'') = 1$.

$S=\{\boldsymbol{p}_1,..,\boldsymbol{p}_k\}$ encodes a proof if the resolutions encoded by points of S are sufficient to derive an empty clause.

Tests $\boldsymbol{x}_1,..,\boldsymbol{x}_m$ encode a proof if they are extracted from a set of points $\{\boldsymbol{p}_1,..,\boldsymbol{p}_k\}$, $k \geq m$ encoding a proof.

Checking if a Set of Points Encodes a Proof

Given CNF formula $F = F^N \wedge z$ and $S = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_k\}$.

1. Find clauses C', C'' of F encoded by points $\boldsymbol{p}'$, $\boldsymbol{p}''$ of S and producing a new resolvent C
2. If C', C'' do not exist, **STOP**.
3. If the resolvent C is an empty clause, **STOP**.
4. Add C to F. Go to step 1.

STOP: S does not encode a proof

STOP: S encodes a proof

Small Complete Test Sets

A proof of $(F^N \wedge z) \equiv 0$ of k resolutions
is encoded by $2*k$ points (at most)

A complete set of $\leq 2*k$ tests for checking $N \equiv 0$.

N with 1000 inputs: 10^6 tests instead of 2^{1000}

Boundary Points

Given a CNF formula F and literal l, an unsatisfying assignment to $Vars(F)$ is an l-boundary point $\boldsymbol{p}$ iff

$$(C(\boldsymbol{p})=0) \wedge (C \in F) \Rightarrow l \in C$$

Let $\boldsymbol{p}'$ be an l-boundary point of F and $\boldsymbol{p}'' = flip(\boldsymbol{p}', l)$. Then $F(\boldsymbol{p}'') = 1$ or $\boldsymbol{p}''$ is an $\sim l$-boundary point of F.

If $\boldsymbol{p}'$ is an l-boundary point of F and F is unsatisfiable, in any proof there is a resolution on literals l and $\sim l$ producing resolvent C such that $C(\boldsymbol{p}') = 0$ and $C(\boldsymbol{p}'') = 0$.

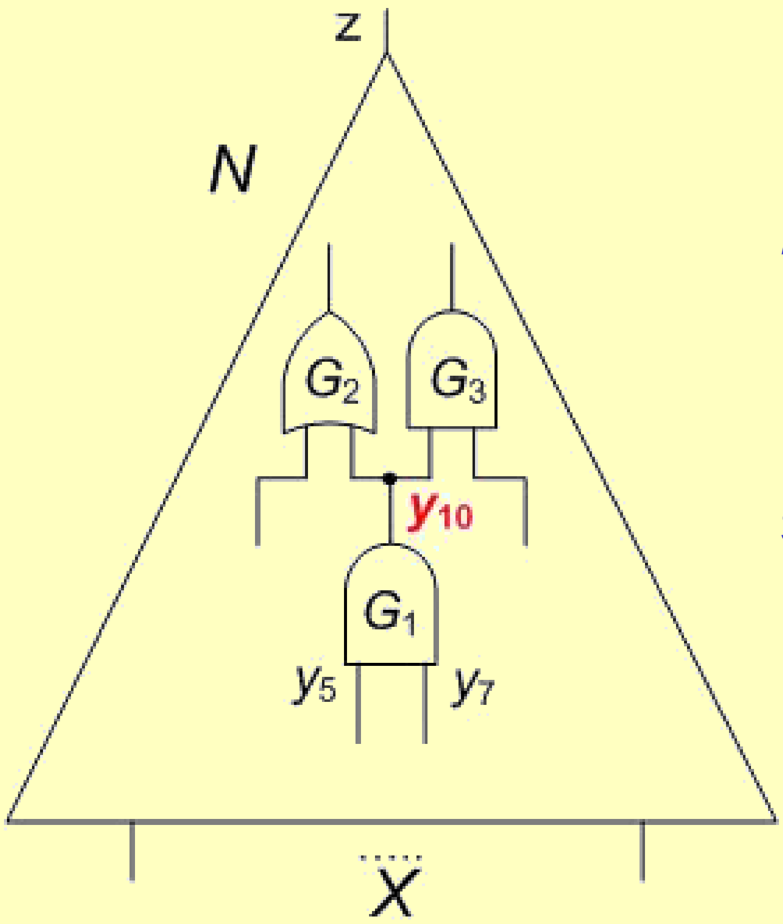
Boundary Points Encode Mandatory Fragments of a Proof

If $\boldsymbol{p'}$ and $\boldsymbol{p''}$ are l and $\sim l$ boundary points of F such that $Hamming\_distance(\boldsymbol{p'},\boldsymbol{p''})=1$, they encode a mandatory resolution on literals l and $\sim l$

After adding the resolvent C of this resolution to F, $\boldsymbol{p'}$ is not an l-boundary point of F (because $C(\boldsymbol{p'}) = 0$).

Finding an l-boundary point reduces to checking the satisfiability of $F \setminus$ {clauses with l}.

Example of a Boundary Point



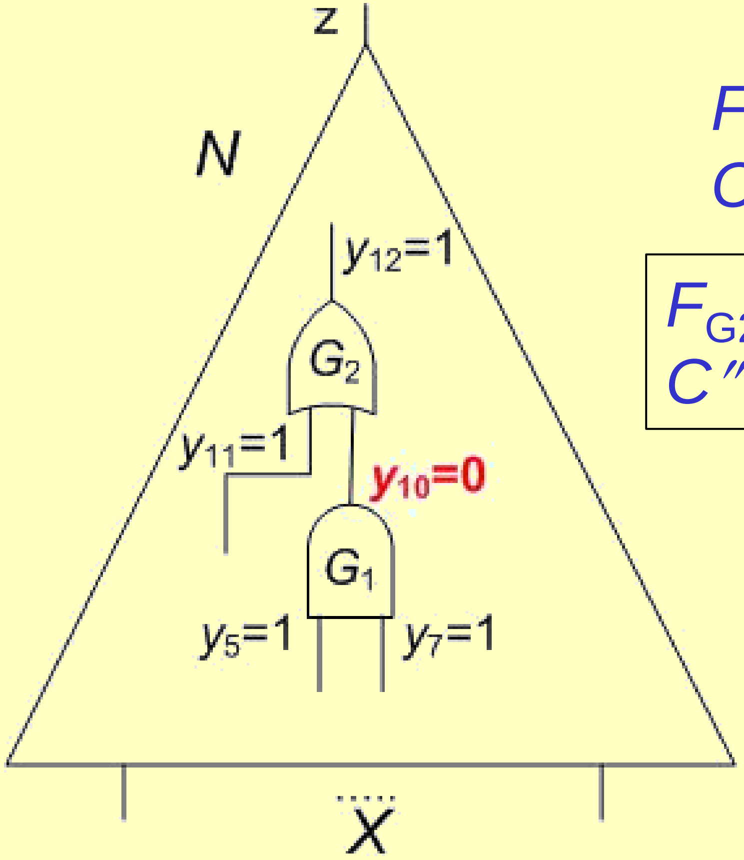
$F = F_N \wedge z$, $F_{G1} \subseteq F_N$

$F_{G1} = C \wedge (y_5 \vee \sim y_{10}) \wedge (y_7 \vee \sim y_{10})$,
$C = \sim y_5 \vee \sim y_7 \vee y_{10}$

Let $\boldsymbol{p}$=(..,y_5=1,..,y_7 =1,….,y_{10}=0,..) satisfies all clauses of F but C

$\boldsymbol{p}$ is an y_{10}-boundary point of F.
(It is also $\sim y_5$-boundary and $\sim y_7$-boundary point)

Test Extracted from a Boundary Point



$F_{G1} = C' \wedge (y_5 \vee \sim y_{10}) \wedge (y_7 \vee \sim y_{10})$,
$C' = \sim y_5 \vee \sim y_7 \vee y_{10}$

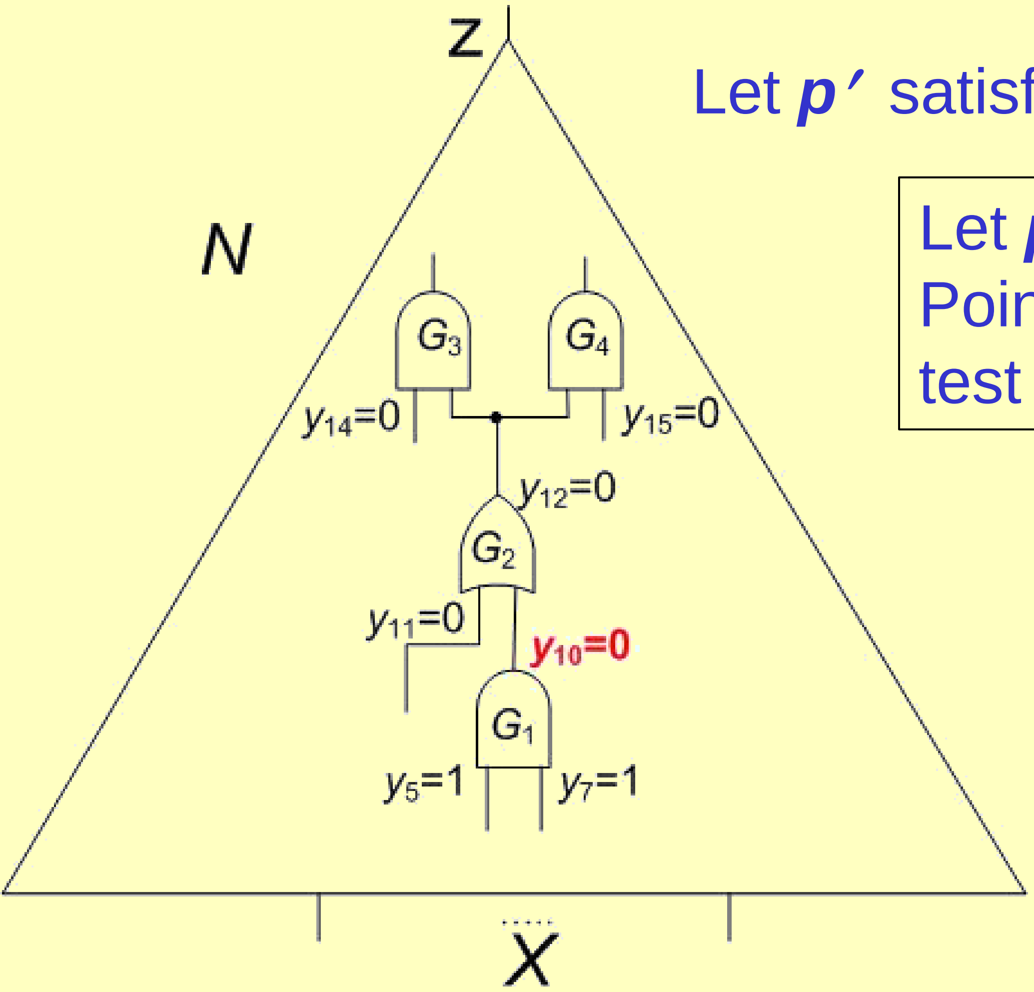
$F_{G2} = (y_{10} \vee y_{11} \vee \sim y_{12}) \wedge C'' \wedge (\sim y_{11} \vee y_{12})$,
$C'' = \sim y_{10} \vee y_{12}$

Let $\boldsymbol{p}'$ satisfy the clauses of F but C'.

Point $\boldsymbol{p}'' = flip(\boldsymbol{p}', y_{10})$ satisfies F

$\boldsymbol{p}' = (\boldsymbol{x}, \boldsymbol{y}', 1)$, $\boldsymbol{p}'' = (\boldsymbol{x}, \boldsymbol{y}'', 1)$
Application of $\boldsymbol{x}$ to N produces the trace $(\boldsymbol{y}'', 1)$ of $\boldsymbol{p}''$

Non-trivial Case



Let $\boldsymbol{p'}$ satisfy all clauses of F but C of F_{G1}.

Let $\boldsymbol{p''} = flip(\boldsymbol{p'}, y_{10})$.
Point $\boldsymbol{p''}$ falsifies F_{G2} but the test $\boldsymbol{x}$ of $\boldsymbol{p''}$ is good.

After applying $\boldsymbol{x}$ to N,
y_{10} : $0 \rightarrow 1$, y_{12} : $0 \rightarrow 1$.
Other change is blocked by $y_{14}=0$, $y_{15}=0$. So $N(\boldsymbol{x}) = 1$

$\boldsymbol{p'} = (\boldsymbol{x}, \boldsymbol{y'}, 1)$, $\boldsymbol{p''} = (\boldsymbol{x}, \boldsymbol{y''}, 1)$
Application of $\boldsymbol{x}$ to N:
trace is different from $(\boldsymbol{y''}, 1)$

Extracting Tests from Boundary Points

Circuit N , CNF formula $F = F^N \wedge z$, Is $N(\boldsymbol{x})=1$?

1. Generate an l-boundary point $\boldsymbol{p}_i$ of F
2. Extract test $\boldsymbol{x}_i$ from $\boldsymbol{p}_i$
3. $N(\boldsymbol{x}_i)=1$? If so, stop.
4. Add to F a resolvent on l and $\sim l$ mandated by $\boldsymbol{p}_i$
5. Go to step 1 to generate $\boldsymbol{p}_{i+1}$

$\forall$ satisfiable $F^N \wedge z$, $\exists$ a bnd. pnt. $\boldsymbol{p}$ such that $N(\boldsymbol{x})=1$

Summary

- Introduction
- Test generation algorithm based on TTPE
- Experimental results and conclusion

Boundary Points and Stuck-at Fault Model

- Tests detecting suck-at faults is a special case of tests extracted from boundary points

- These points are computed for the formula describing equivalence checking of two identical copies of the circuit under test

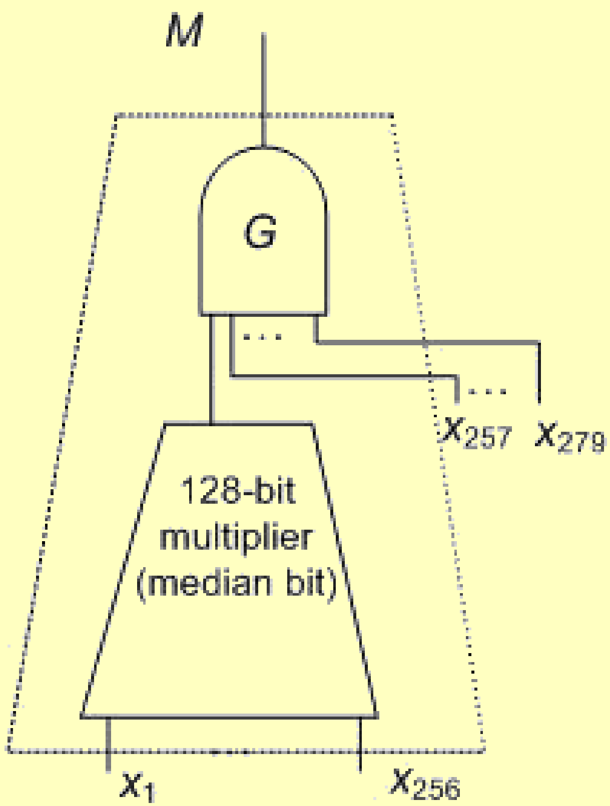
- The success of the stuck-at model shows that tests extracted from boundary points computed for a circuit *N* can be used for a modified version of *N*

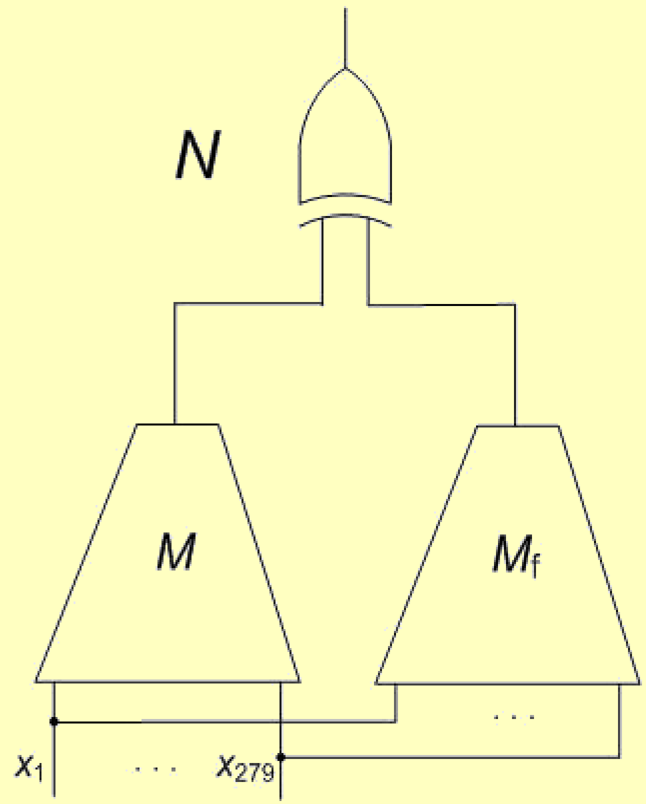
The Reasons for the Success of the Stuck-at Model

- One may say that the stuck-at fault model is a trick to produce tests extracted from boundary points

- The success of the stuck-at fault model may have little to do with its proximity to real faults

Faults in Arithmetic Components





G is a 24-input AND gate

Comparing SAT, Random Tests and Tests from Boundary Points

- 17 faults. Random tests failed (10^6 per fault)
- Reused tests generated for previous faults

	Precosat	Tests from boundary pnts	
	time (s.)	time (s.)	#tests
total	54, 115	2,919	562
average	3,183	172	33
median	935	8	3

- Precosat is a winner of SAT-2009 competition
- Used Precosat for finding boundary points too

Some Concluding Remarks

- TTPE is not a trick. There is a deep relation between proofs and tests.
- Many ways to use TTPE (e.g. encoding mandatory parts of a proof).
- TTPE for more expressive logics (describing sequential circuits and software).