Outline: ATPG for stuck at faults

- Deterministic, Fault-Oriented ATG
  - D-Algorithm
  - PODEM
  - FAN
- Random ATG
  - Weighted random
  - RAPS
- Test Compaction

D-Algorithm

- Roth (1966) proposed a D-algebra and a deterministic ATG algorithm.
  - D: good value 1 / faulty value 0
  - D: good value 0 / faulty value 1

5-valued algebra:

<table>
<thead>
<tr>
<th>V_G / V_F</th>
<th>0</th>
<th>1</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>0/0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1/1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1/0</td>
<td>D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0/1</td>
<td>D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-/X,X/-</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
D-Algorithm: 5-Valued Operations

\[
\begin{array}{c|ccc|}
A & B & C \\
\hline
0 & - & 0 \\
- & 0 & 0 \\
1 & 1 & 1 \\
1 & X & X \\
D & 1 & D \\
1 & D & D \\
1 & D & D \\
D & X & X \\
D & D & D \\
\end{array}
\]

\[
\begin{array}{cccc}
& 0 & 1 & D \\
\hline
0 & 0 & 0 & 0 \\
1 & 0 & 1 & D \\
D & 0 & D & 0 \\
X & 0 & X & X \\
\end{array}
\]

Fanout-Free Circuit

- Test for \( m \ s-a-1 \) in a fanout-free circuit
  - \textbf{begin}
    - set all lines to \( X \);
    - Justify \((m, 0)\); # activate the fault
    - Propagate \((m, D)\);
  - \textbf{end}

- no conflicts with implications or justifications
Justify

<table>
<thead>
<tr>
<th>Justify d=0</th>
<th>Justify d=1</th>
</tr>
</thead>
<tbody>
<tr>
<td>a b c</td>
<td>a b c</td>
</tr>
<tr>
<td>d=0</td>
<td>d=1</td>
</tr>
</tbody>
</table>
| select one i of {a,b,c} & justify i=0 | justify (a=1)
| justify (a=0) | justify (b=1)
| justify (b=0) | justify (c=1)
| justify (c=0) | justify (a=0) |
| a d=0 | a d=1 |
| justify (a=1) | justify (b=0) & justify (c=0) |
| a b c | a b c |
| d=0 | d=1 |
| justify (a=1) | justify (b=1) & justify (c=1) |
| justify (b=0) | justify (a=0) |
| justify (c=0) | justify (b=0) |
| d=0 | d=1 |
| select one i of {a,b,c} & justify i=1 | justify (c=0) |

Select one i ∈ {a,b,c}

Propagate (a, D)

Propagate D on line a

- justify (b=1) and justify (c=1)
- justify (b=0) and justify (c=0)
- select one path i ∈ {a_1, a_2}
  Propagate (i, D)
Test Generation

Set $g=1$
justify $(g=1)$
justify $(a=1),\ justify\ (f=1)$
justify $(d=0),\ \rightarrow\ justify\ (c=0)$
propagate $(g,\ D)$
justify $(h,\ 0)$
   $\rightarrow$ select one, justify $(e=0)\ \rightarrow\ justify\ (c=0)$

Test Generation

- What if we make “wrong” selection (decision)?
- What if justify $(a=1)$ fails?
- What if propagate fails?

Set $g=0$
justify $(g=0)$
select $f\ \rightarrow\ justify\ (f=0)$
justify $(d=1)$
justify $(e=1)$
propagate $(g,\ D)$
justify $(h,\ 0)$
select $e$
justify $(e=0)$
justify $(c=0)$  $\times$
Backtracking

- Conflicts can occur in circuits with fanout and reconvergence.
  - If a decision causes inconsistency, then we need to backtrack.
  - A backtracking strategy is simply a systematic exploration of all decisions (choices).
- Conflict/inconsistency/contradiction
  - An already-assigned value is different from the value implied by the last decision.
- Bounding conditions
  - There is no D left in the circuit.
  - The fault is not excited.
  - Lookahead indicates that a D cannot propagate.

Choice in D-Propagation
General Test Generation Algorithm

- Procedure `Imply_and_check()` implies and checks for inconsistency.
- Procedure `Solve()` is a generic branch-and-bound procedure.
  - **AND-OR search strategy.**

\[
\begin{align*}
\text{set of problems that must be solved} & \{a, b, c\} \text{ AND} \\
\text{solve a} & \quad \text{choice } \{a_1 \text{ OR} a_2 \text{ OR} a_3\} \\
\text{solve a}_1 & \quad \text{solve a}_2 \\
\text{AND} \{x, y\} & \quad \text{solve a}_3 \\
\text{Failure} & \quad \text{Failure} \\
\text{Success} & \quad \text{Success} \\
\text{Success} & \quad \text{Success}
\end{align*}
\]