EECS 579: Test Generation 2

Recap: Basic Concepts

- D notation for errors: D = 1 in fault-free circuit; D = 0 in faulty circuit
- All D/D’ values refer to the same (potential) fault
- Roth’s D-Algorithm formalizes ATPG via path sensitization in terms of signal values {0,1,X,D,D’}
- Key steps (procedures) of D-Algorithm:
  - Propagation
  - Justification
  - Implication
  - Backtracking

D-Algorithm

This ATPG method formalizes path sensitization approach in terms of signal vectors over {0,1,X,D,D’}

Summary

- Initialize all signal values to a “test cube” tc of X’s
- Select a “D-cube of failure” producing D/D’ on line L of gate G.
- Propagation: Select a “propagation D-cube” for G; combine with tc to produce a new tc, thus driving the “D frontier” forward to a new L. Repeat until a primary output is reached
- Justification: Select values to assign to unspecified inputs of gates whose outputs are specified. Iterate backwards until enough lines are justified. The resulting tc defines the required test.
- Implication: To speed up the computation, assign unique signal values that are forced (implied) by any step of the algorithm.
- Backtracking: Backtrack to the most recent decision point and select an alternative decision until none remain
### D-Algorithm: Example 1

**Diagram:**

```
  b
 /\  
 a  c
 \//
  e
```

**Decision Table:**

<table>
<thead>
<tr>
<th>Decision</th>
<th>Implication</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>$B = D'$</td>
<td>$E = D', A = 0, a = 0$</td>
<td>Activate fault</td>
</tr>
<tr>
<td>$b = 1$</td>
<td>$C = D'$</td>
<td>Propagate via $C$</td>
</tr>
<tr>
<td>$F = 0$</td>
<td>$z = D'$</td>
<td>End of D-drive</td>
</tr>
<tr>
<td>$H = 0$</td>
<td>$e = 0$</td>
<td>Justify $F$</td>
</tr>
<tr>
<td>$c = 0$</td>
<td></td>
<td>Justify $A$</td>
</tr>
</tbody>
</table>

Test: $abce = 0100$
The D-Algorithm

\[
\text{D-alg()}
\begin{align*}
\text{begin} & \\
& \text{if } \text{Imply and check()} = \text{FAILURE then return FAILURE} \\
& \text{if (error not at PO) then} \\
& \quad \text{begin} \\
& \quad \quad \text{if } D\text{-frontier} = \emptyset \text{ then return FAILURE} \\
& \quad \quad \text{repeat} \\
& \quad \quad \quad \text{begin} \\
& \quad \quad \quad \quad \text{select an untried gate } (G) \text{ from } D\text{-frontier} \\
& \quad \quad \quad \quad \text{c = controlling value of } G \\
& \quad \quad \quad \quad \text{assign } c \text{ to every input of } G \text{ with value } x \\
& \quad \quad \quad \quad \text{if } D\text{-alg()} = \text{SUCCESS then return SUCCESS} \\
& \quad \quad \quad \text{end} \\
& \quad \quad \text{until all gates from } D\text{-frontier} \text{ have been tried} \\
& \quad \text{return FAILURE} \\
& \text{end} \\
& \text{/* error propagated to a PO */} \\
& \text{if } J\text{-frontier} = \emptyset \text{ then return SUCCESS} \\
& \text{select a gate } (G) \text{ from the } J\text{-frontier} \\
& \text{c = controlling value of } G \\
& \text{repeat} \\
& \quad \text{begin} \\
& \quad \quad \text{select an input } (j) \text{ of } G \text{ with value } x \\
& \quad \quad \text{assign } c \text{ to } j \\
& \quad \quad \text{if } D\text{-alg()} = \text{SUCCESS then return SUCCESS} \\
& \quad \quad \text{assign } c \text{ to } j \quad \text{/* reverse decision */} \\
& \quad \text{end} \\
& \text{until all inputs of } G \text{ are specified} \\
& \text{return FAILURE} \\
\end{align*}
\]
Example 2

<table>
<thead>
<tr>
<th>Decisions</th>
<th>Implications</th>
</tr>
</thead>
<tbody>
<tr>
<td>d=0</td>
<td>Activate the fault</td>
</tr>
<tr>
<td>i=1</td>
<td>Unique D-drive through g</td>
</tr>
<tr>
<td>j=1</td>
<td>Propagate through i</td>
</tr>
<tr>
<td>k=0</td>
<td>Propagate through n</td>
</tr>
<tr>
<td>l=1</td>
<td>Contradiction</td>
</tr>
<tr>
<td>m=1</td>
<td>Propagate through k</td>
</tr>
<tr>
<td>n=1</td>
<td>Propagate through n</td>
</tr>
<tr>
<td>o=1</td>
<td>Contradiction</td>
</tr>
<tr>
<td>p=1</td>
<td>Propagate through m</td>
</tr>
</tbody>
</table>

Example 3

<table>
<thead>
<tr>
<th>Decision</th>
<th>Implication</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>x_0</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>x_1</td>
<td>b</td>
<td></td>
</tr>
<tr>
<td>x_2</td>
<td>c</td>
<td></td>
</tr>
<tr>
<td>x_3</td>
<td>d</td>
<td></td>
</tr>
</tbody>
</table>
**Backtracking**

- This is the main iteration control method used by the D-algorithm.
- If a decision step fails, undo the most recent decision and choose a different one if available, e.g.:
  - Select a different propagation path
  - Select a different D/D' on the D-frontier
  - Select a different gate justification value

**Complications**

- The D-algorithm performs too much backtracking in some circuits. All possible choices may be tried in the worst case.
- Some faults require multiple-path sensitization
- PODEM addresses these problems by using a very different signal assignment method and eliminating explicit justification

---

**Example 3**

![Diagram of a circuit with decision implication actions](image)

- **Decision**
- **Implication**
- **Action**
D-Algorithm

Excessive backtracking occurs in certain circuit types

\[ 2^{n-1} \text{ justifying values} \]

PODEM

- PODEM = Path-Oriented DEcision Making [Goel 1981]
- Like the D algorithm, this is a circuit-based, fault-oriented ATPG algorithm
- Signal values are explicitly assigned at the primary inputs only; other values are computed by implication.
- Justification is not needed
- Backtracking means reassigning primary inputs when a contradiction occurs; “implicit enumeration” technique
- A simple “backtrace” heuristic is used to select the next primary input line and the value to assign to it
- Many extensions exist with more complex heuristics, for example, the FAN algorithm
**Simplified PODEM**

- Input assignment with implication only; no backtrace used

**Example 1**

<table>
<thead>
<tr>
<th>Decision</th>
<th>Implication</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a = 1$</td>
<td>$A = 1, B = 1$</td>
<td>Contradiction at fault Backtrack</td>
</tr>
<tr>
<td>$a = 0$</td>
<td>$A = 0, B = \overline{D}, E = \overline{D}, H = 0, F = 0, C = \overline{D}, z = \overline{D}$</td>
<td>Contradiction at fault Backtrack</td>
</tr>
<tr>
<td>$b = 1$</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>$c = 1$</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>$e = 1$</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>$e = 0$</td>
<td>-------------</td>
<td>---------</td>
</tr>
</tbody>
</table>

**PODEM**

**Summary**

- Backtracing selects a primary input $L$ and a 0/1 value $v$ for $L$
- All implications of each assignment are determined
- If $D/D'$ is implied on a primary output, a test has been found
- Otherwise a new $v$ or a new primary $L$ is selected

**Initial Objectives**

- The first initial objective $IO_0$ tries to apply $v = D/D'$ to the fault site
- Subsequent initial objectives try to drive $D/D'$ to a primary output

**Backtracing**

- For each $IO_j$, a path is traced backwards to a primary input via a series of “current” objectives.
- Current objectives are selected by heuristics that try to maximize the chances of satisfying $IO_j$. 
PODEM

PODEM()
{
    if (D or \overline{D} at PO) return(SUCCESS);
    if (test not possible) return(FAILURE);
    (k, v_k) = objective();
    (j, v_j) = backtrace(k, v_k);
    imply(j, v_j); /* j is a PI */
    /* Visit next level in decision tree */
    if (PODEM() == SUCCESS) return(SUCCESS);
    /* Reverse decision */
    imply(j, \overline{v_j});
    if (PODEM() == SUCCESS) return(SUCCESS);
    /* No solution this way, reset input to x */
    imply(j, x);
    return(FAILURE);
}