EEDG/CE 6303: Testing and Testable Design

Mehrdad Nourani

Dept. of ECE
Univ. of Texas at Dallas

Session 03

Test Generation for Combinational Circuits

Generic Test Generation Algorithm

1. Read the circuit under test (CUT)
2. Preprocessing
   a. Identify static indirect implications
   b. Compute testability metrics
3. For each target fault in the fault list
   a. Insert the target fault by using the appropriate description of the behavior of the faulty element
   b. Initialize circuit lines
      - In five-valued system, assign X_5 to all lines
      - In six-valued system, assign X_6 to all lines in transitive fanout of fault site and χ_6 to all other lines
      - In sixteen-valued system
        + Assign {$\underline{0}$, $\underline{1}$, D, $\overline{D}$} to all lines in transitive fanout of the fault site
        + Assign {$\underline{0}$, $\underline{1}$} to all the other lines
        + Perform forward implication starting at each output of the faulty circuit element

Generic Test Generation Algorithm (cont.)

c. Identify test generation sub-tasks (TGSTs)
   - Analyze values at primary outputs of CUT and outputs of the faulty circuit element
   - Analyze gates in the D-frontier D
   - Analyze the set of unjustified lines U
   - Determine, whether
     + Test generation is complete – print vector, manage fault list, go to Step 3 and process next fault
     + Conflict, i.e., not possible to generate a vector for target by further specifying currently assigned vector (values) – initiate **backtrack**
     + Continue search by carrying out a TGST
       1. FEE – fault-effect excitation,
       2. FEP – fault-effect propagation, or
       3. Justification

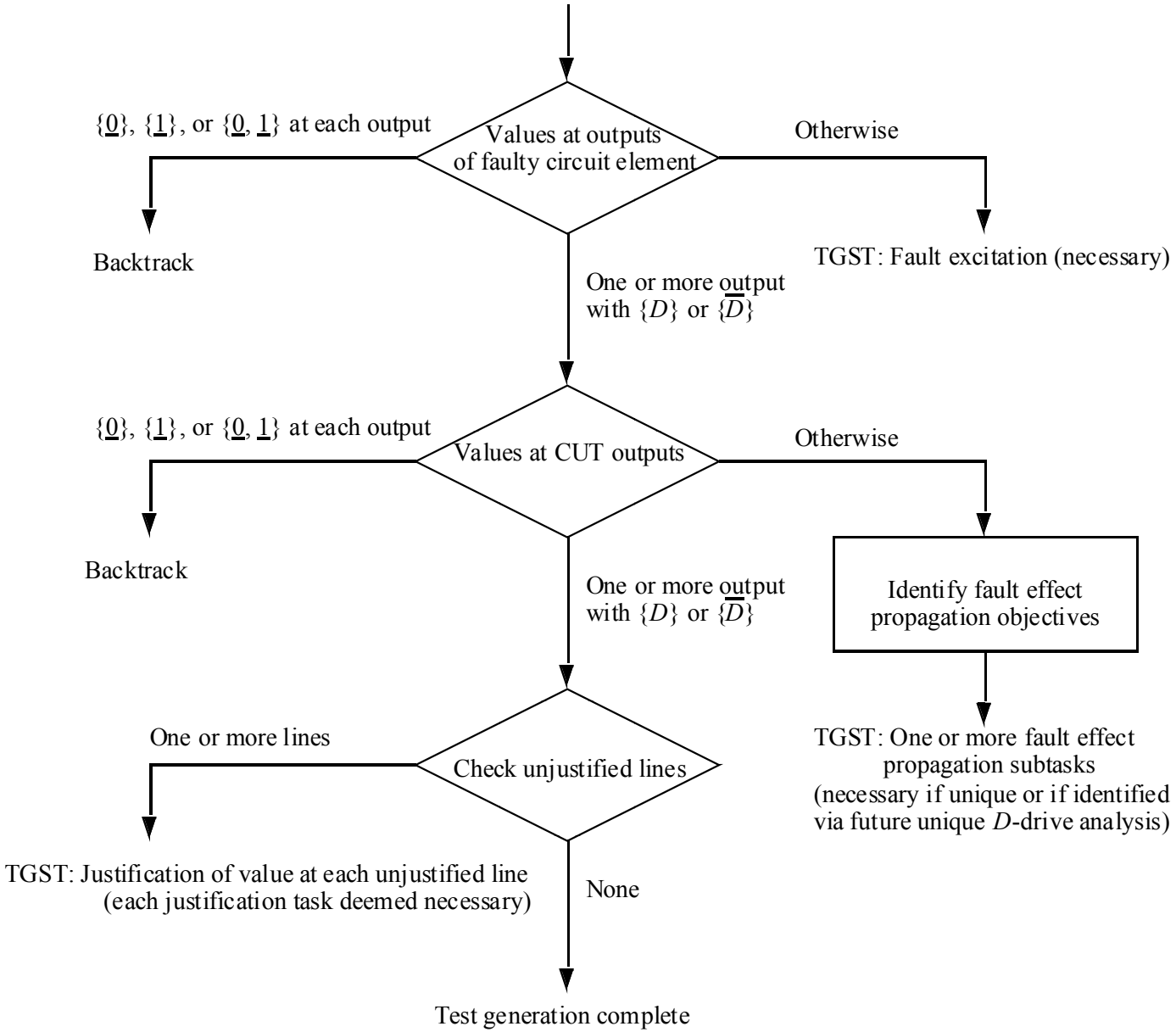
d. Identify a value assignment to accomplish the selected TGST

e. Assign selected value
   - Save the state of the test generation: e.g., values at circuit lines, untried alternatives TGSTs and/or value assignments
   - Assign selected value
   - Perform implications
     + If successful, continue
     + Else, backtrack

Identification of TGSTs

- Backtrack
  - Find the most recent value assignment at which an untried alternative existed
  - Restore the state of the test generation algorithm to prior to that value assignment
  - Make an alternative TGST/value assignment and try it, starting at Step 3d or 3e
- Assignment of each logic value by test generator followed by implication that
  - Either returns CONFLICT and initiates backtrack
  - Else, returns SUCCESS after updating
    - Values at circuit lines
    - D-frontier D
    - List of unjustified lines U
- Subsequently, values at circuit lines, D, and U are analyzed to identify TGSTs with following possible outcomes
  - Fault-effect excitation (FEE)
  - Fault-effect propagation (FEP)
  - Justification
  - Backtrack
    - FEE impossible
    - FEP impossible
  - Test generation complete

Identification of TGSTs



Test Generation Algorithms

D-Algorithm

- Initialization
- Identification of TGSTs
  0. If impossible, BACKTRACK; if complete, STOP
  1. Fault-effect excitation (FEE) – **necessary**
  2. Fault-effect propagation (FEP)
     - Each gate in D-frontier D an **alternative**
  3. Justification
     - Every gate in U – **necessary**
- Select a TGST
  - Identify local value assignment for selected TGST
    - FEE – values at **outputs** of faulty circuit element
    - FEP – value at the **output** of gate selected for FEP
    - Justification – values at **inputs** of gate selected for justification

D-Algorithm

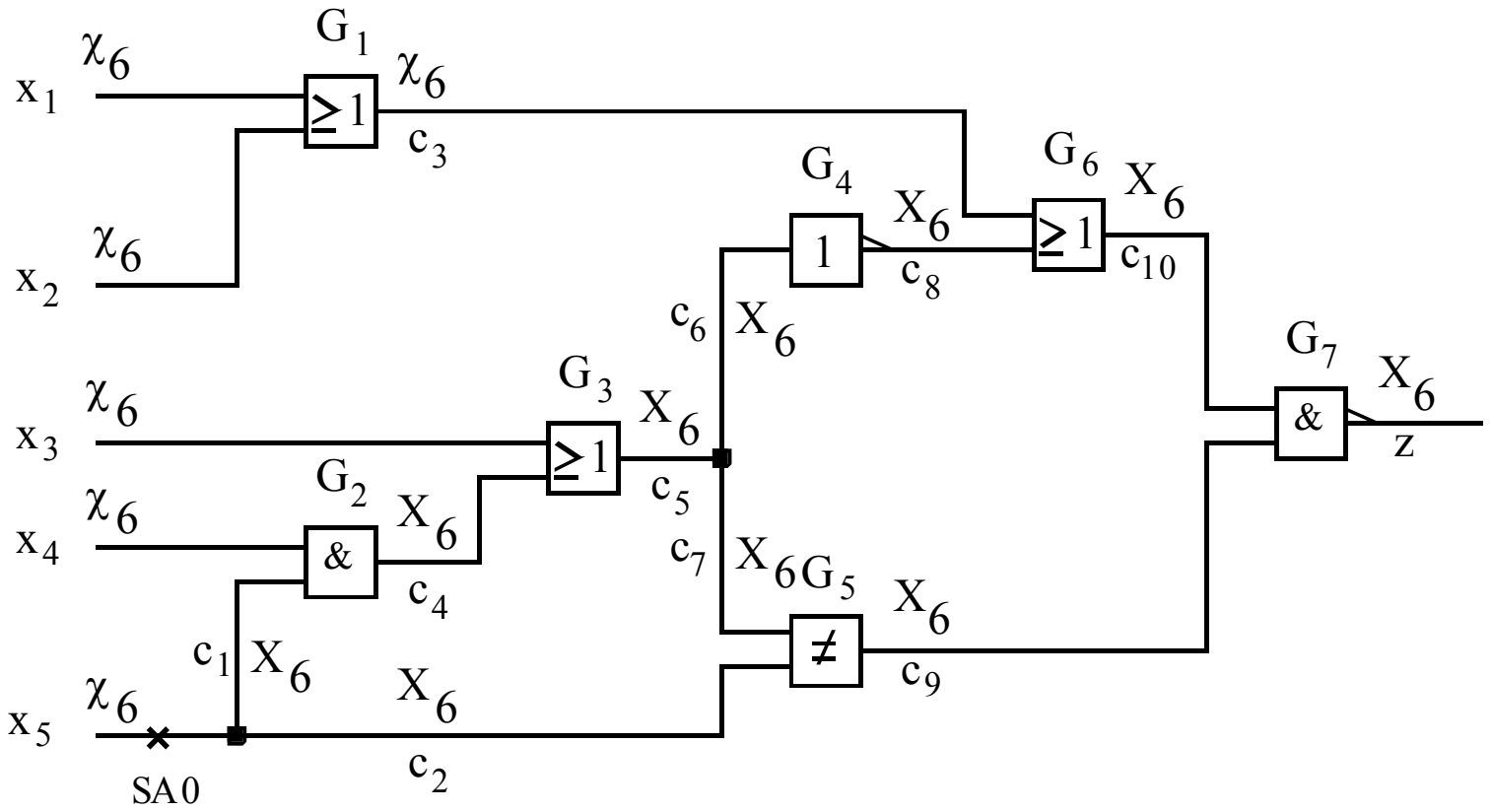
- Value assignment
  - —Store untried alternative TGSTs and untried alternative value assignment for selected TGST
  - —Store the values at all circuit lines
  - —Make selected local value assignment (VA)
  - —Perform implication – if CONFLICT, then BACKTRACK
- BACKTRACK
  - —Find most recently selected VA or TGST for which one (or more) untried alternative exists
  - —Restore values to **before** above assignment was made
  - —Select an untried alternative TGST and/or untried local VA and repeat the value assignment step

D-Algorithm

- Identification of local value assignment for FEE
  - Value at outputs of faulty circuit element
  - D_6 or $\overline{D}_6$; in some cases both
  - Intersect current values at inputs and outputs of faulty circuit element with **fault excitation cubes** of the version of the circuit element with the target fault
  - If all cubes have D_6 at an output, local VA at that output is D_6
  - If all cubes have $\overline{D}_6$ at the output, local VA at that output is $\overline{D}_6$
  - If some cubes have D_6 at the output and other have $\overline{D}$, two alternative VAs at the output, namely D_6 and $\overline{D}_6$

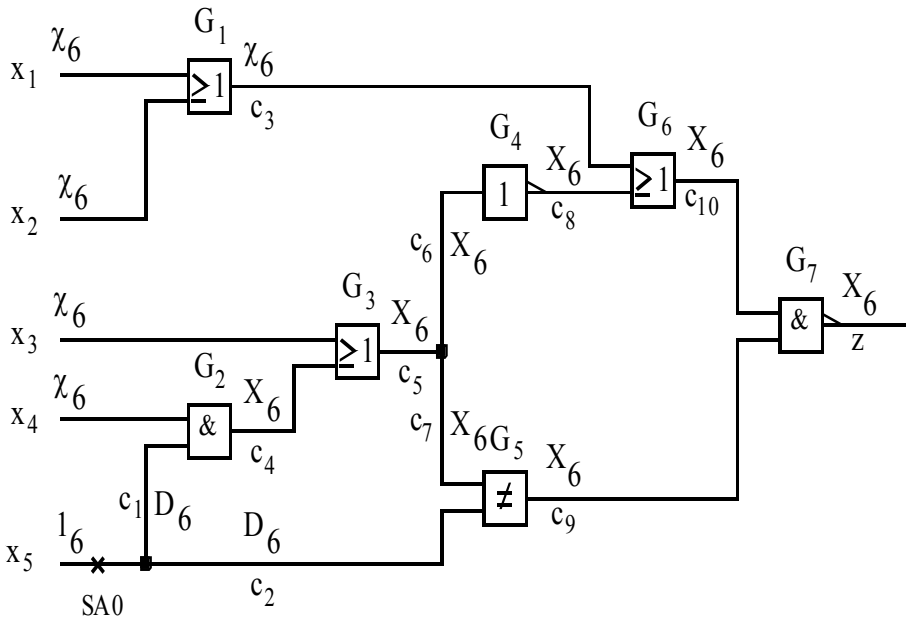
D-Algorithm – Example (0)

- Target fault: x_5 SA0
- After initialization



D-Algorithm – Example (1)

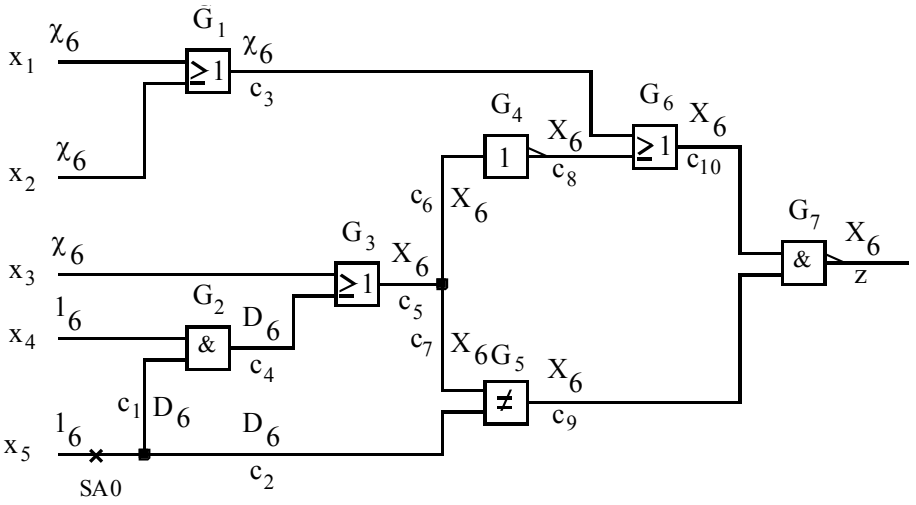
- Round 1: TGST identified FEE (no untried alternative)
  - Local VA: D_6 at c_1 and D_6 at c_2 (no untried alternatives)
  - Implication



  - D = {G_2, G_5}
  - U = { }

D-Algorithm – Example (2)

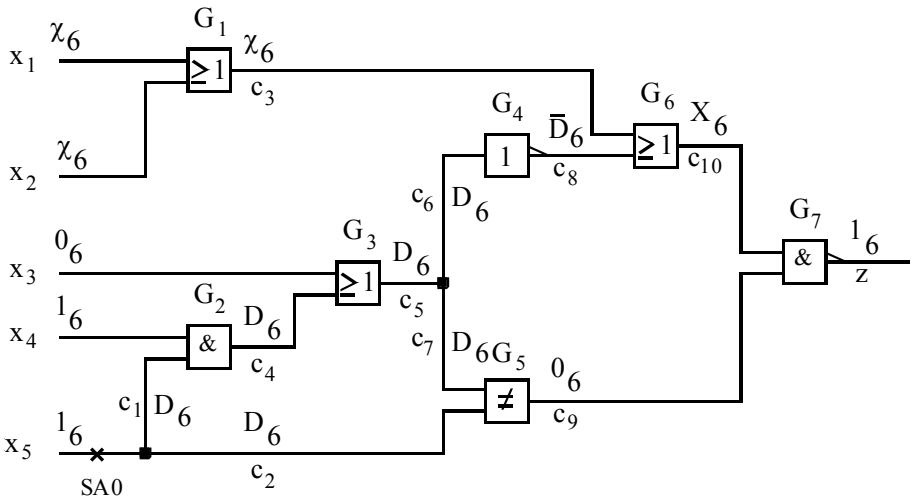
- Round 2: TGST selected: FEP via G_2 (untried alternative – FEP via G_5)
  - Local VA identified: D_6 at c_4 (no untried alternative)
  - Save the values at all lines
  - Save the untried alternative TGST (FEP via G_5)
  - Assign identified local VA
  - Implication



  - D = {G_3, G_5}
  - U = { }

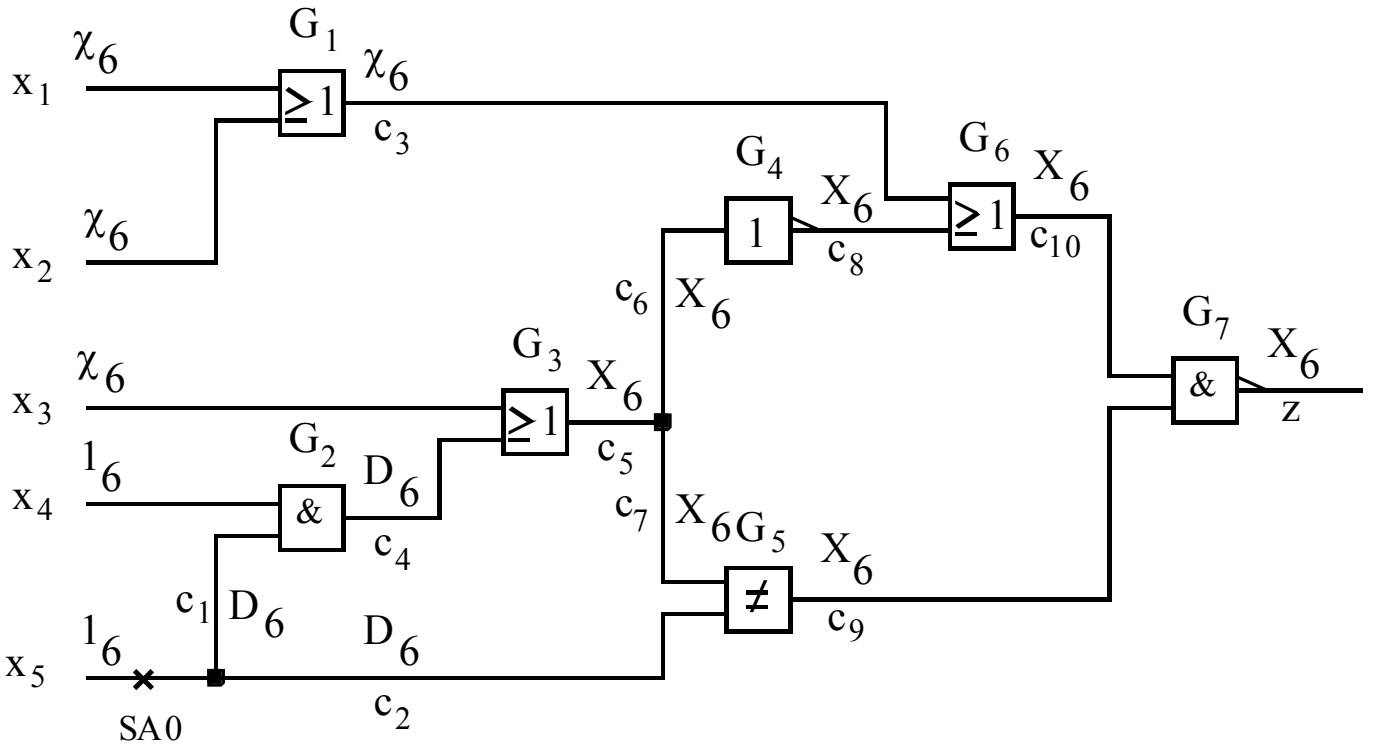
D-Algorithm – Example (3)

- Round 3: TGST selected: FEP via G_3 (untried alternative – FEP via G_5)
  - Local VA identified: D_6 at c_5 (no untried alternatives)
  - Save the values at all lines
  - Save the untried alternative TGST (FEP via G_5)
  - Assign identified local VA
  - Implication
  - D = {G_6}
  - U = { }



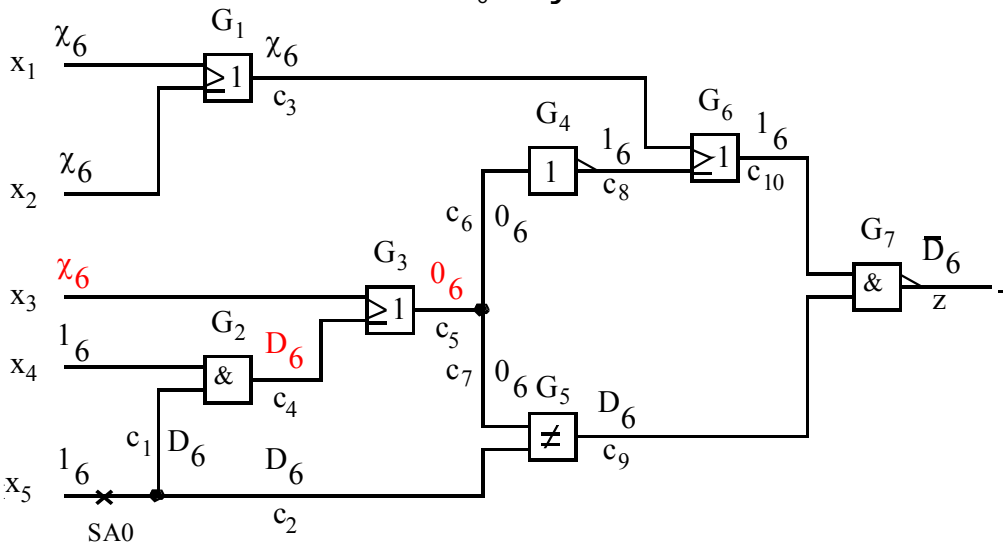
D-Algorithm – Example (4)

- Round 4: TGST identified – Backtrack (since x-path check from the only gate in D failed)
  - Find the most recent round where an untried alternative exists
  - Restore values prior to the corresponding value assignment



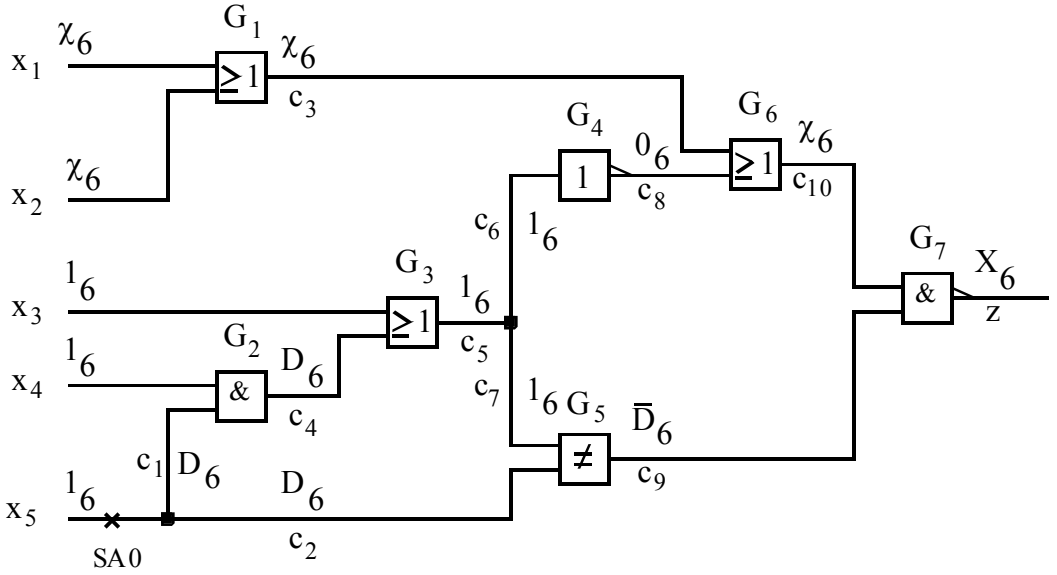
D-Algorithm – Example (5)

- Round 5: Since following backtrack
  - Do **not** identify TGST, but
  - **Select** one of the untried alternatives
  - TGST selected: FEP via G_5 (no alternatives exist)
  - Local VAs identified
    - Alternative 1: D_6 at c_9
    - Alternative 2: $\overline{D}_6$ at c_9
  - Select the first (D_6 at c_9)
  - Save the untried local VAs (untried local VA $\overline{D}_6$ at c_9)
  - Assign selected local VA
  - Implication
    - Returns CONFLICT
    - Backtrack to most recent round with untried alternative
      - Round 5 with untried local VA $\overline{D}_6$ at c_9



D-Algorithm – Example (6)

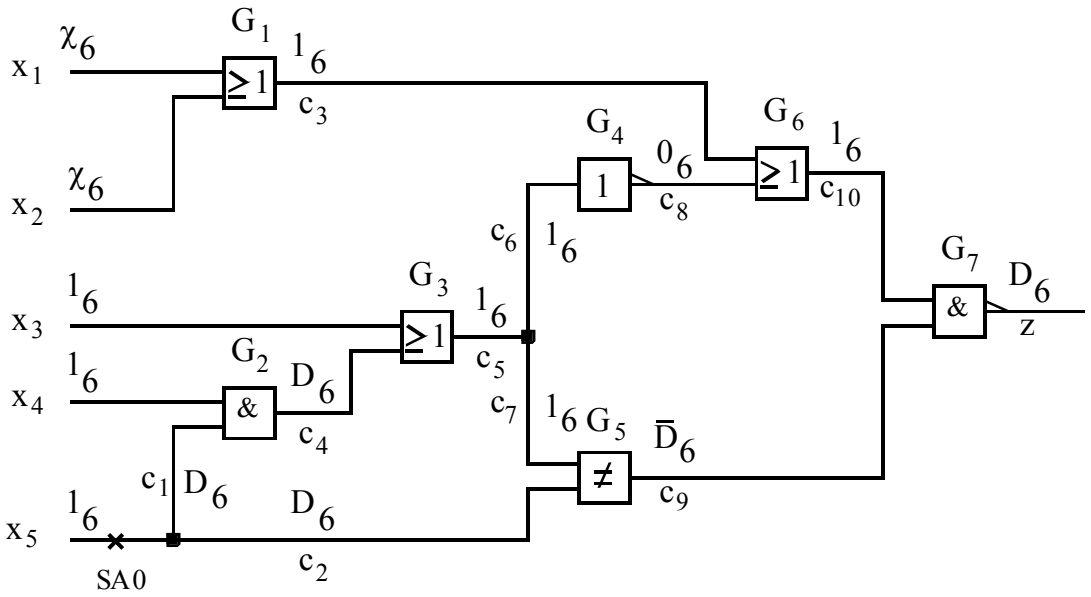
- Round 6: Since backtrack in previous round, untried alternative used
  - Use previously untried local VA, namely $\overline{D}_6$ at c_9 (no untried alternative left)
  - Assign above local VA
  - Implication



  - D = {G_7}
  - U = { }

D-Algorithm – Example (7)

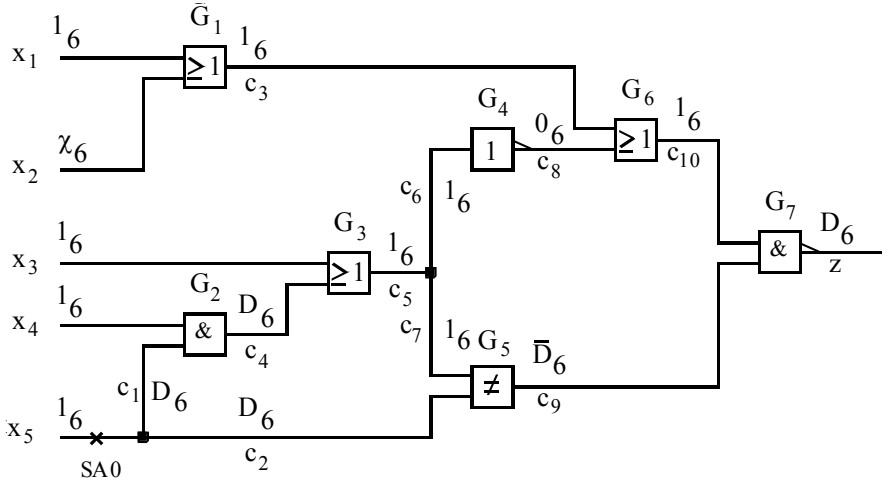
- Round 7: TGST identified: FEP via G_7 (no untried alternative TGST)
  - Local VAs identified - D_6 at z
  - Assign D_6 at z (no untried alternative)
  - Implication



- D = { }
- U = {G_1}

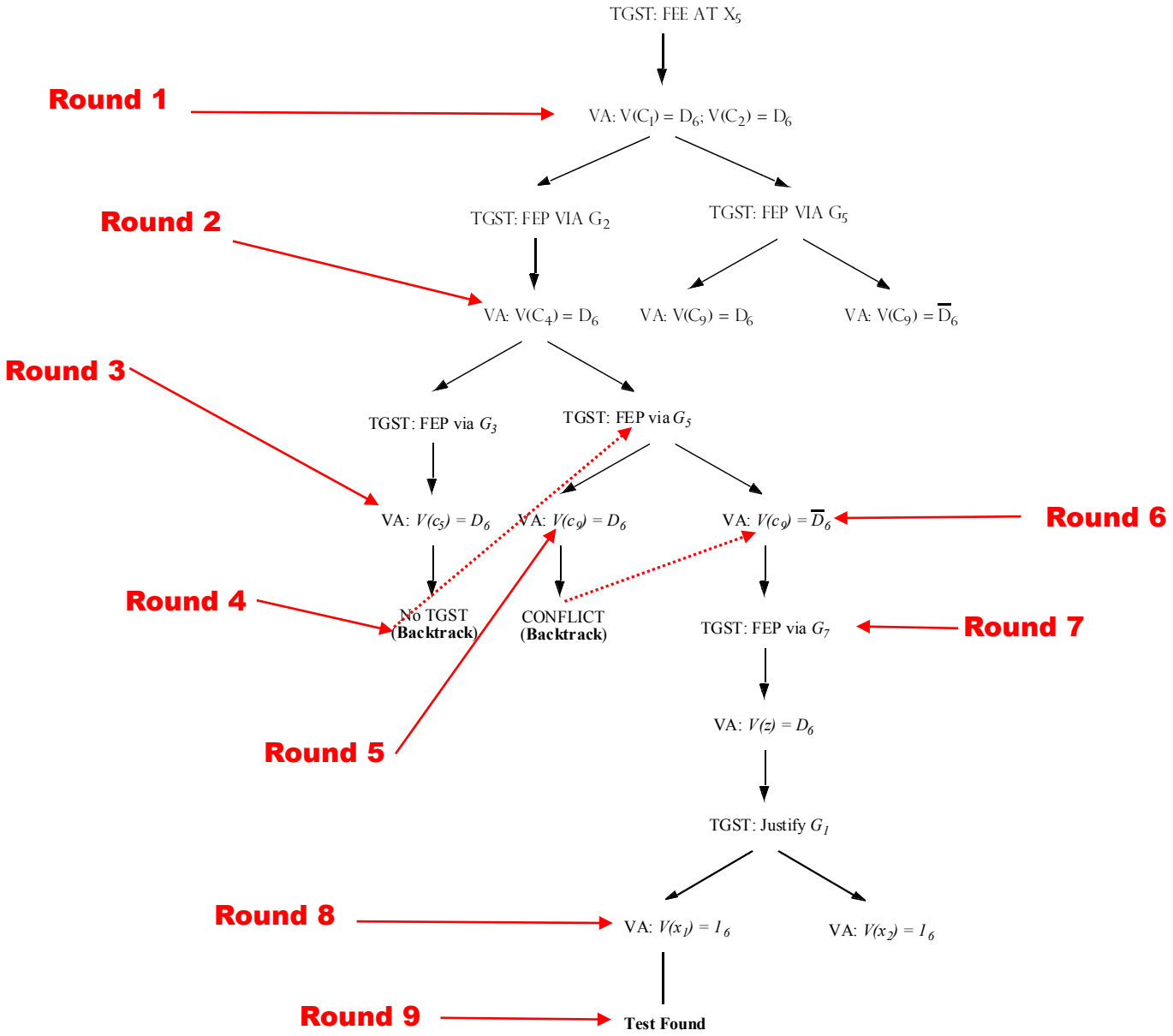
D-Algorithm – Example (8)

- Round 8: TGST identified: Justify G_1 (**necessary**)
  - Local VAs identified
    - Alternative 1: 1_6 at x_1
    - Alternative 2: 1_6 at x_2
  - Select the first (untried alternative local VA 1_6 at x_2)
  - Save the values at all lines
  - Save untried alternative (local VA 1_6 at x_2)
  - Assign selected local VA (1_6 at x_1)
  - Implication
  - D = { }
  - U = { }



- Round 9: TGST identified – test generation complete

D-Algorithm – Search Process



Path-Oriented Decision Making (PODEM)

- A test for a fault in an n-input circuit is one of 2^n possible vectors
- Hence test generation can be formulated as a search in the space of input vectors by
  - **Assigning values only at primary inputs**
  - Predictable maximum search space size
  - No unjustified lines exist during test generation
  - Forward implication only
  - State of test generator
    - Values at primary inputs
    - List of untried alternative value assignments at primary inputs

Basic Steps in PODEM

1. Assign binary value to unassigned PI
2. Determine implications of all PIs
3. Test Generated? If so, done.
4. Test possible with more assigned PIs? If maybe, go to Step 1
5. Is there untried combination of values on assigned PIs? If not, exit: untestable fault
6. Set untried combination of values on assigned PIs using objectives and backtrace. Then, go to Step 2

PODEM Algorithm

- Initialization – similar to D-algorithm
- Identification of TGSTs – similar to D-algorithm, except
  - Only FEE and FEP TGSTs identified, i.e., no justification TGSTs
- Local value assignment identification – similar to D-algorithm except
  - One alternative selected and others discarded, i.e., not recorded as untried
- Value identified and selected above **not** assigned (unless at a primary input)
  - Instead, this value is treated as an **objective** identified as (line, value)
  - **Backtrace** procedure used to translate the objective to a value assignment at a primary input

PODEM Algorithm

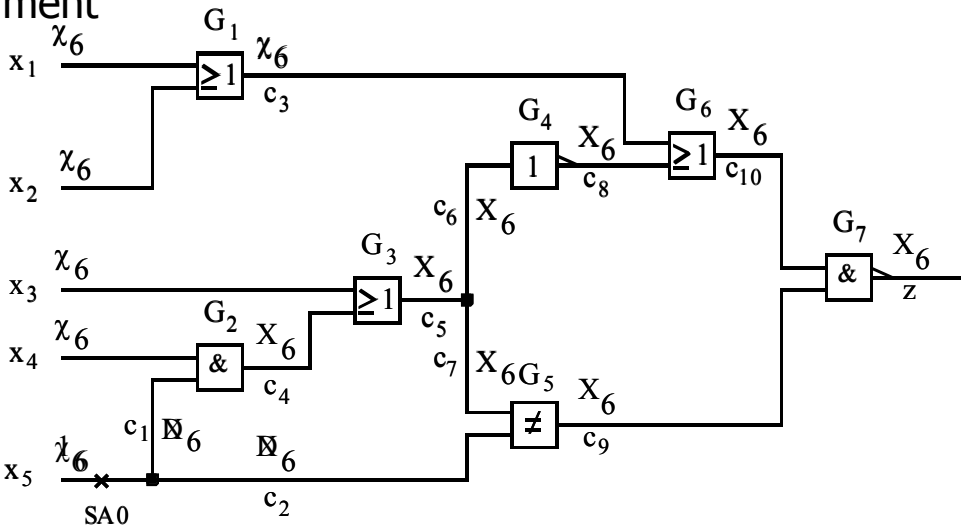
- Backtrace:
  - An objective (c_i, V) is translated to an objective (c_j, V'), where c_j is a fanin of c_i
    - Let c_i be output of gate G. Current values at inputs of G and value V at output combined into a cube C
    - C intersected with all cubes in cube-cover of G
      - Even cubes where some elements are incompatible, the intersection is noted for inputs that are compatible
      - Let c_j be an input at which for one of the cubes, intersection gives a value V' that is different from the current value at c_j
    - c_j is deemed **visited** and (c_j, V') is adopted as the next objective
  - Above process is repeated, until an assignment (x_l, V*) is found at a primary input

PODEM – Example (1)

- Example: SA0 at x_5
- Round 1
  - Initialization
  - Identify TGST – FEE identified
  - Compute alternative local VAs for this TGST
  - Select 1_6 at x_5 as initial objective and backtrace
    - Backtrace terminates with (x_5, 1_6)
  - Value assignment
    - Save (x_5, 0_6) as untried assignment
    - Save values at all inputs
    - Assign 1_6 at x_5
    - Perform implication

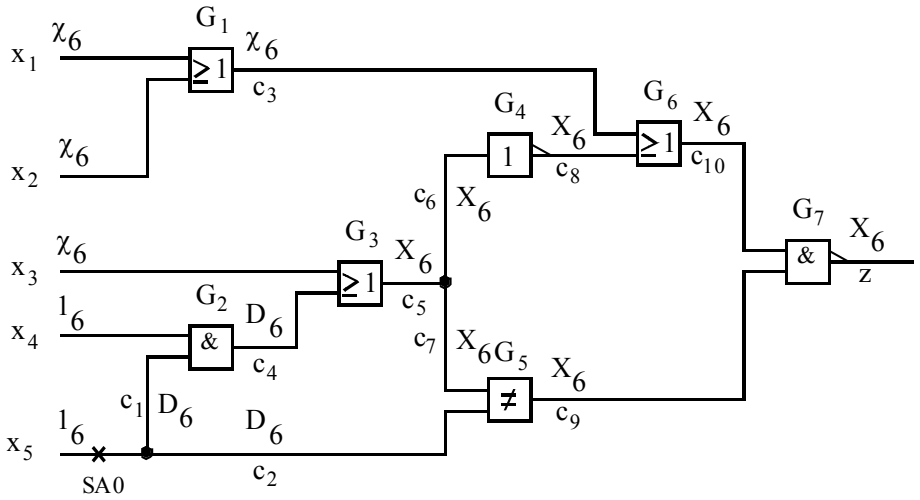
x_5	c_1	c_2
1_6	D_6	D_6

- D = {G_2, G_5}
- U = { } (**always**)



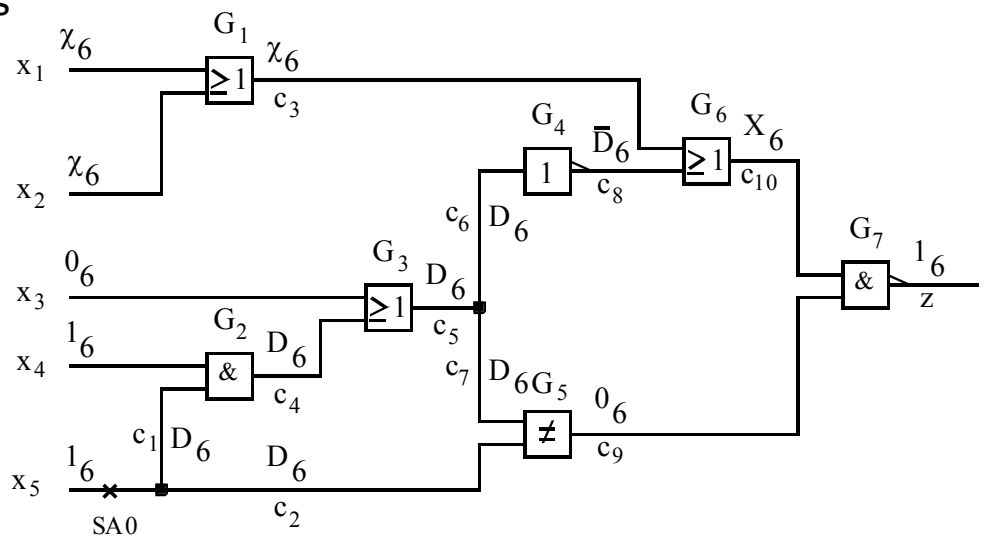
PODEM – Example (2)

- Round 2
  - TGST identification
    - Alternative 1: FEP via G_2
    - Alternative 2: FEP via G_5
    - Select one, say FEP via G_2 (need **not** store the other alternative)
  - Identify local VAs for the selected TGST, namely FEP via G_2
    - Identify initial objective (x_4, 1_6)
  - Execute backtrace
    - Since x_4 is a primary input, final objective is (x_4, 1_6)
  - Value assignment
    - Save (x_4, 0_6) as an untried alternative VA
    - Save the values at all primary inputs
    - Assign 1_6 at x_4
    - Perform implications
    - D = {G_3, G_5}



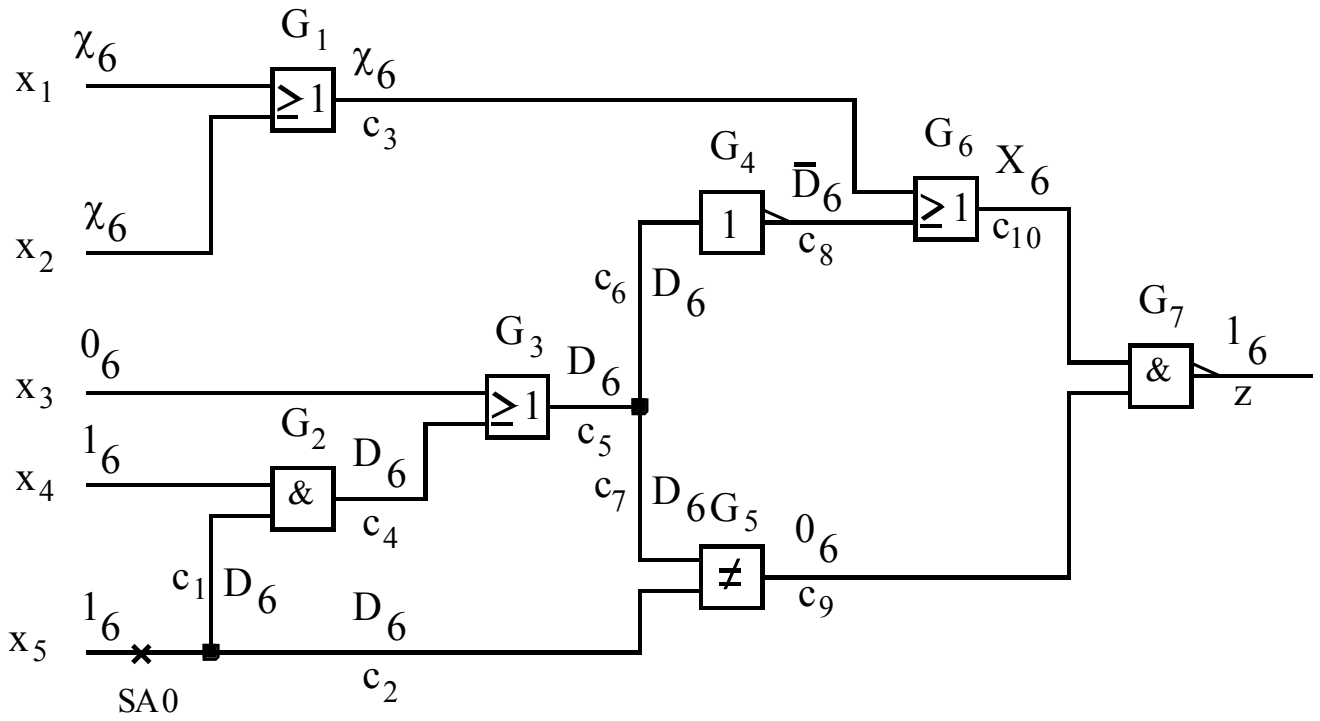
PODEM – Example (3)

- Round 3
  - TGST Identified
    - Alternative 1: FEP via G_3
    - Alternative 2; FEP via G_5
    - Select one, say FEP via G_3 (need **not** store the other)
  - Identify local VA for FEP via G_3
    - $(x_3, 0_6)$
  - Backtrace initial objective $(x_3, 0_6)$ to obtain final objective $(x_3, 0_6)$
  - Value assignment
    - Save $(x_3, 1_6)$ as untried alternative
    - Save value at all primary inputs
    - Assign 0_6 at x_3
    - Perform implication



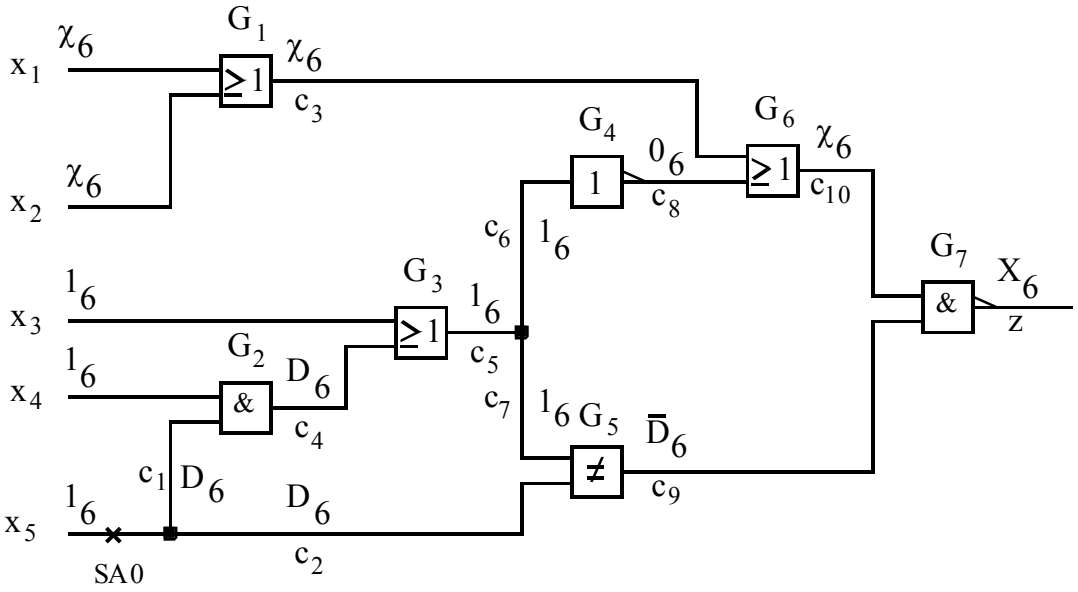
PODEM – Example (4)

- Round 4
  - TGST identified – none, hence BACKTRACK
  - Backtrack
    - Find the most recent primary input assignment for which an untried alternative exists
      - At Round 3, untried alternative (x_3, 1_6) exists
    - Backtrack to values at the beginning of Round 3



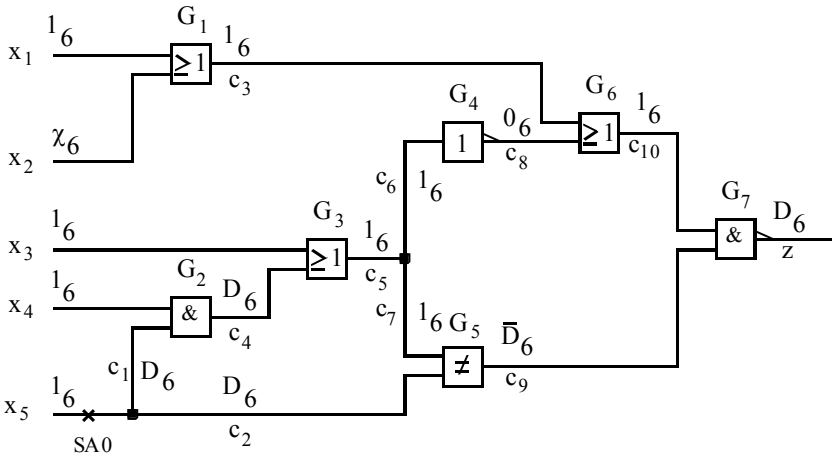
PODEM – Example (5)

- Round 5
  - Since following backtrack, used the untried alternative identified in previous round, namely (x_3, 1_6)
  - Value assignment
    - Untried alternative – none
    - Assign 1_6 at x_3
    - Implication
  - D = {G_7}



PODEM – Example (6)

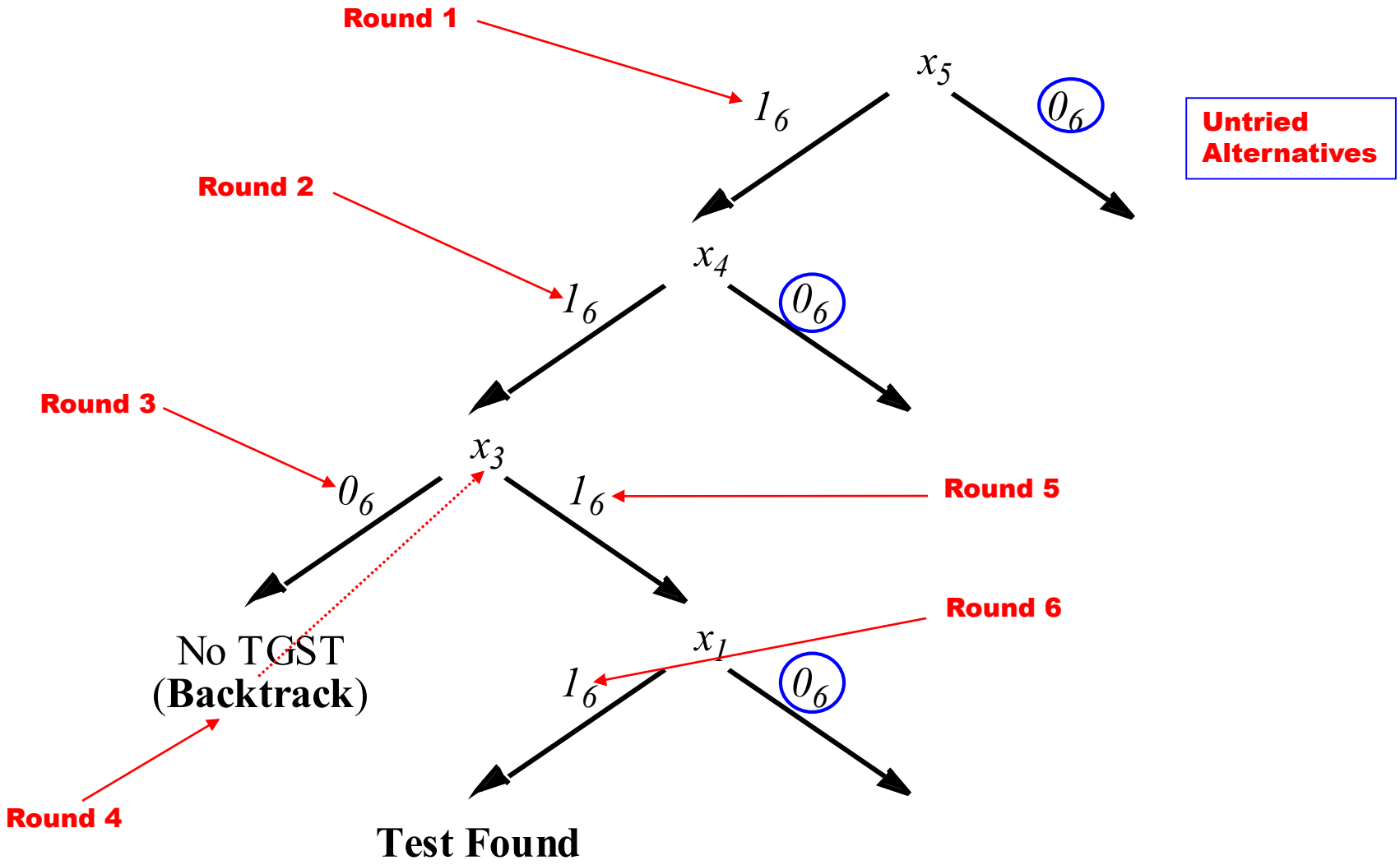
- Round 6
  - Identify TGST FEP via G_7
  - Identify local VAs – (c_{10}, 1_6)
  - Backtrace with (c_{10}, 1_6) as initial objective to obtain final objective (x_1, 1_6)
  - Value assignment
    - Note (x_1, 0_6) as untried alternative
    - Save values at all primary inputs
    - Assign 1_6 at x_1
    - Implication



  - In the next round, TGST identification discovers that test generation is complete

PODEM – Search Process

- PODEM search tree for the running example



D-Algorithm vs. PODEM

- D-algorithm and PODEM differ in
  - The structure of the search tree
    - In D-algorithm, alternative TGSTs and VAs
    - In PODEM, value assignments at primary inputs
      - Predictable maximum complexity for PODEM is 2^n
  - The state of test generation algorithm
    - In D-algorithm, typically
      - Values at **all** circuit lines
      - Search tree
    - In PODEM
      - Values at primary inputs
      - Search tree
    - However, in PODEM, backtrack must be followed by implication on all circuit lines where incompletely/completely specified values existed
      - Hence, memory to store state of test generation reduced at the expense of increased computational complexity
    - If similar memory vs. computational complexity tradeoff used in D-algorithm
      - Only values at unjustified lines and primary inputs need to be stored

D-Algorithm vs. PODEM (cont.)

- D-algorithm and PODEM differ in
  - Ability to take advantage of known necessary conditions
  - Since most necessary conditions are value assignments at internal circuit lines
    - D-algorithm can assign all such values
    - PODEM can only select one value at a time and backtrace, to eventually assign a value at a primary input
      - Backtrace with one necessary condition as an objective
        - Ignores other known necessary conditions
        - May select a value assignment that is neither necessary nor sufficient for the condition selected as the original backtrace objective

Accelerating Test Generation

- There are many approaches in the literature trying to speed up the test generation process.
- Two main categories
  - **Deterministic**: Guaranteed to reduce search complexity
  - **Heuristic**: Likely to reduce search complexity
  - Even a deterministic technique may not reduce overall complexity if the complexity required to implement the technique **exceeds** the reduction in search complexity