Testing Digital Systems I

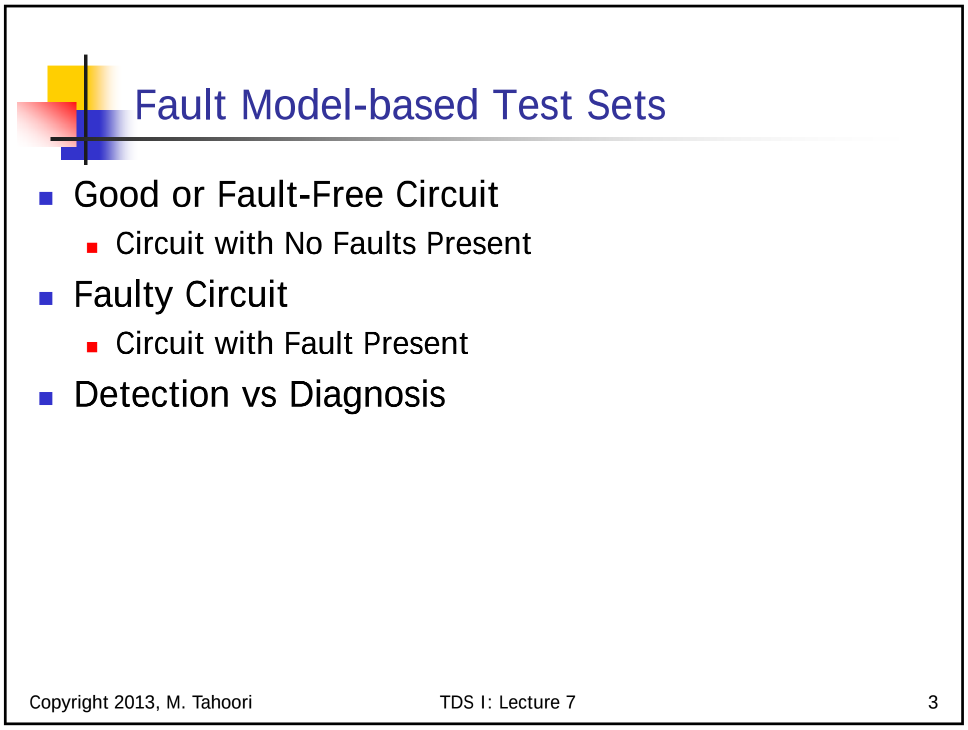
Lecture 7:

Boolean Testing Using Fault Models

Instructor: M. Tahoori

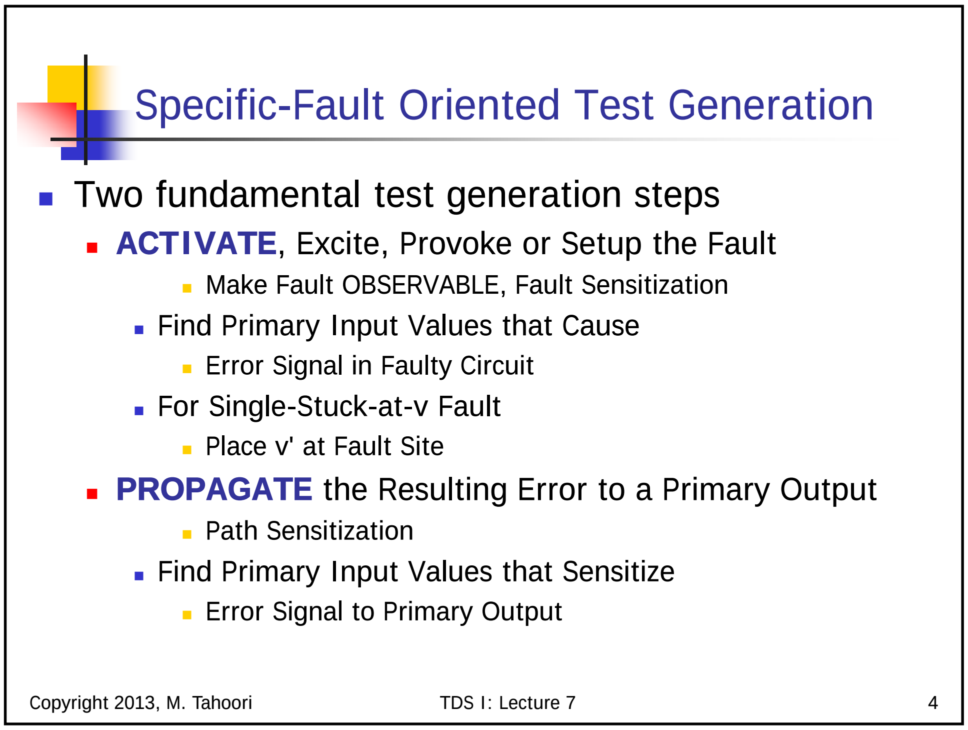
This Lecture

- Specific Fault Objective — Target Fault
  - Boolean
    - Algebraic and Boolean Difference
  - Path Tracing
    - D Algorithm, PODEM, Fan



Fault Model-based Test Sets

- Good or Fault-Free Circuit
  - Circuit with No Faults Present
- Faulty Circuit
  - Circuit with Fault Present
- Detection vs Diagnosis

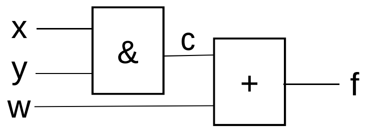


Specific-Fault Oriented Test Generation

- Two fundamental test generation steps
  - **ACTIVATE**, Excite, Provoke or Setup the Fault
      - Make Fault OBSERVABLE, Fault Sensitization
    - Find Primary Input Values that Cause
      - Error Signal in Faulty Circuit
    - For Single-Stuck-at-v Fault
      - Place v' at Fault Site
  - **PROPAGATE** the Resulting Error to a Primary Output
      - Path Sensitization
    - Find Primary Input Values that Sensitize
      - Error Signal to Primary Output

Specific-Fault Oriented Test Generation

- Example: Test for c/0 is w,x,y = 0,1,1
  - ACTIVATE Fault c/0
    - Set x = y = 1 to make c=1
      - in Fault-free Circuit
  - PROPAGATE Value on c to f
    - Set w =0 to sensitize c to f



Line Justification

- Find Input Assignment to Place Value v on Line g
- Algebraic Approach
  - Find Boolean Function Realized on line g = G(X)
  - Use Prime Implicant of G(X) to Place 1 on g
  - Use Prime Implicate of G(X) to Place 0 on g

- PROPAGATE Error (Fault Effect)
  - Algebraic Approach
    - Use Boolean Difference

Boolean Difference

Boolean Difference

- Shannon expansion
  - A Boolean function $f(X_1, X_2, \dots X_n)$ can be expanded about any variable X_i
  - $f(X_1, X_2, \dots X_n) = X_i' f(X_1,\dots,X_i = 0,\dots X_n) + X_i f(X_1,\dots,X_i = 1,\dots X_n)$
- Boolean Difference of f(X1, X2, ... Xn) with respect to Xi
- Symbol is (partial derivation)

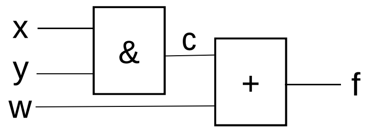
$$\frac{d\, f(X1, Xi, \dots Xn)}{dXi}$$

- Definition is:
- $\dfrac{df}{dXi} = f_{Xi'} \oplus f_{Xi} = f(X1,\dots,Xi = 0,\dots Xn) \oplus f(X1,\dots,Xi = 1,\dots Xn)$

Boolean Difference

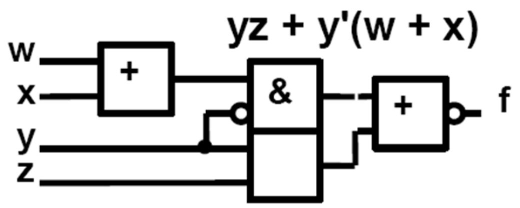
- Example
  - $f = w + xy$,
  - $f_{y'} = w$
  - $f_y = w + x$
  - $\frac{df}{dy} = (w) \oplus (w + x) = w'x$

Boolean Difference



- $\frac{df(x,y,w)}{dy} = 1$
  - for values of w and x for which f depends on y
- $\frac{df(x,y,w)}{dy} = 0$
  - for values of w and x for which f is independent of y
- $\frac{df(w+xy)}{dy} = w'x$
  - $w'x = 1$, for $w=0$, $x=1$
    - When $w = 0$, $x = 1$, $w + xy = y$

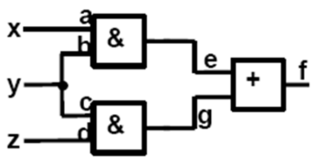
Boolean Difference



- Example
  - df/dy = fy' ⊕ fy
  - = (w + x) ⊕ z
  - = wz' + xz' + w'x'z

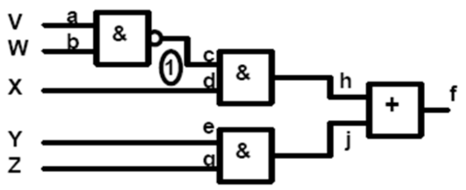
Boolean Difference

- Test pattern generation
  - df/dx = d(xy+yz)/dx = yz ⊕ (y + yz) = yz'
- Test for a/0 is xyz = (110)
  - Set x = 1 to *Provoke* Fault
  - Set y = 1 , z = 0 to *Sensitize* Fault Site to Output



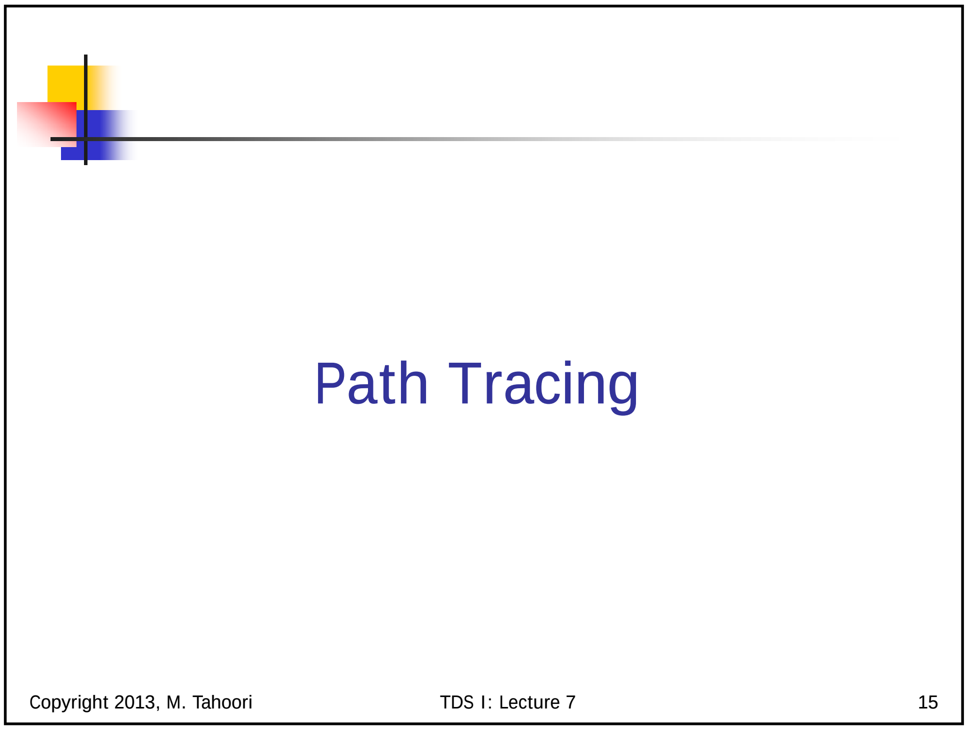
Boolean Difference

- Test pattern generation
  - C/1
  - df/dc = d(cx+yz)/dc = yz ⊕ (x + yz) = x(y' + z) = x(y' + z')
  - To Propagate Fault, Set x = 1, y or z = 0
  - c = v' + w'
  - For c/1 , must set c = 0,
    - so v = w =1

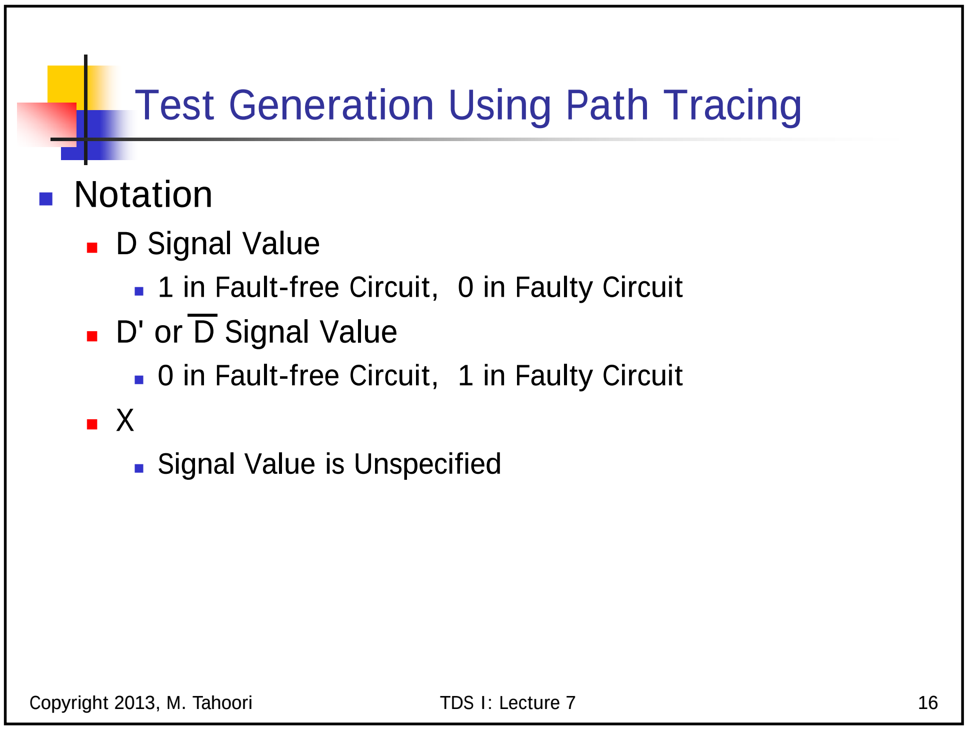


Boolean Difference

- Algebraic Technique to Determine
  - Path Sensitization from Fault Site to Output, or
  - Fault Observability Conditions
- Used Mainly for Theoretical Studies



Path Tracing



Test Generation Using Path Tracing

- Notation
  - D Signal Value
    - 1 in Fault-free Circuit, 0 in Faulty Circuit
  - D' or $\overline{D}$ Signal Value
    - 0 in Fault-free Circuit, 1 in Faulty Circuit
  - X
    - Signal Value is Unspecified

Notation

- Truth Table for AND

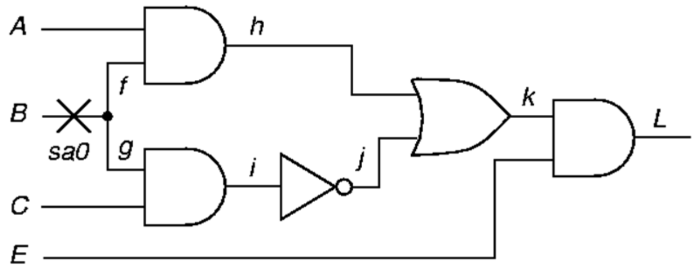
$a \backslash b$	0	1	X	D	$\overline{D}$
0	0	0	0	0	0
1	0	1	X	D	$\overline{D}$
X	0	X	X	X	X
D	0	D	X	D	0
$\overline{D}$	0	$\overline{D}$	X	0	$\overline{D}$

Path Sensitization Method

- Fault Sensitization
  - Force tested node to opposite of fault value
- Fault Propagation (path sensitization)
  - Propagate the effect to one or more POs
- Line Justification
  - Justify internal signal assignments made to activate and sensitize fault
- These three steps may result in conflict
  - Different values are assigned to the same signal
  - Require **backtracking**

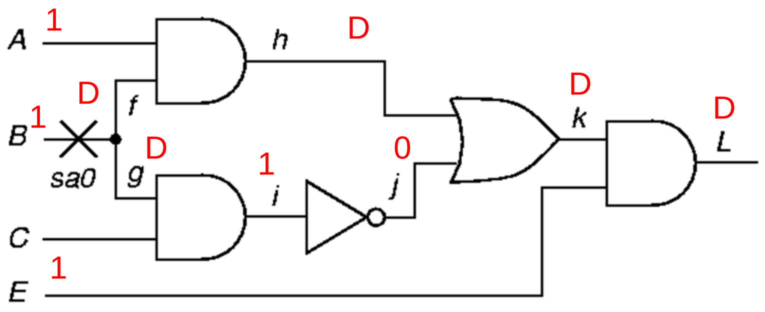
Path Sensitization Method

- Example (B stuck-at 0)
- Fault activation
  - Requires B = 1, f = D, g =D
- Fault propagation
  - Three scenarios are possible
    - paths f – h – k – L , g – i – j – k – L, or both



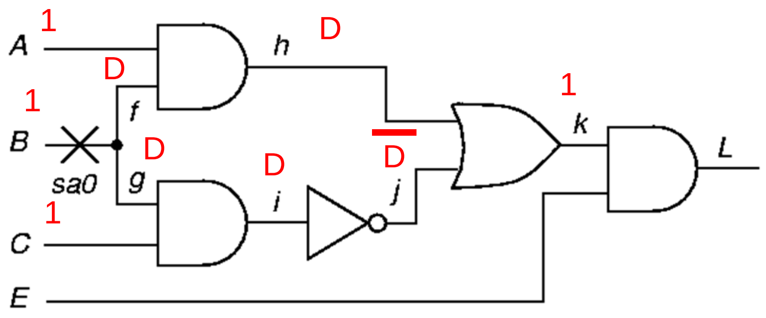
Path Sensitization Method

- Try path f – h – k – L
  - Requires A = 1, j = 0, E = 1
- Blocked at j
  - Since there is no way to justify 1 on i



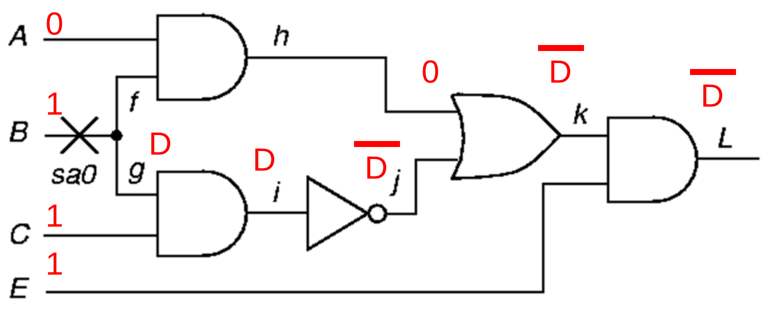
Path Sensitization Method

- Try simultaneous
  - paths f – h – k – L and g – i – j – k – L
- Blocked at k because
  - D-frontier (chain of D or $\overline{D}$) disappears



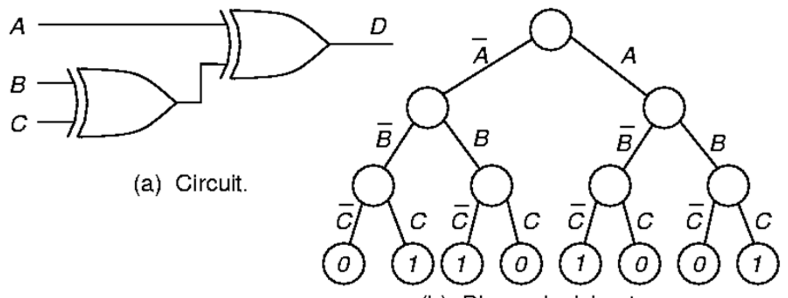
Path Sensitization Method

- Final try: path g – i – j – k – L
  - test found!



Search Space Abstraction

- Binary Decision Tree (BDT)
  - The leaves represent the output of the good machine



(a) Circuit.

(b) Binary decision tree.

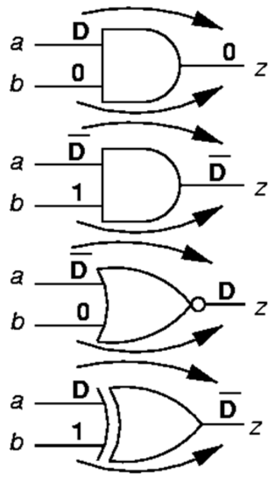
Algorithm Completeness

- All ATPG programs implicitly search BDT
- Definition:
  - Algorithm is complete if it ultimately can search entire binary decision tree, as needed, to generate a test
- Untestable fault
  - No test for it even after entire tree searched
- Combinational circuits only
  - Untestable faults are redundant, showing the presence of unnecessary hardware

ATPG Problem

- Ibarra and Sahni in 1975 showed that ATPG is NP_Complete
  - No polynomial-time algorithm is known
  - Presumed to be exponential
- These ATPG algorithms employ heuristics that
  - Find all necessary signal assignments for a test
    - As early as possible
  - Search as little of the decision space as possible

Forward Implication

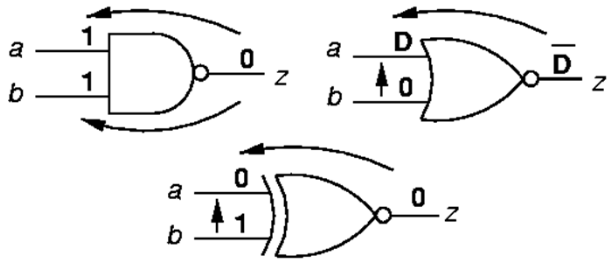


- Results in logic gate inputs that are significantly labeled so that output can be uniquely determined
- Example
  - AND gate forward implication table:

a \ b	0	1	X	D	$\overline{D}$
0	0	0	0	0	0
1	0	1	X	D	$\overline{D}$
X	0	X	X	X	X
D	0	D	X	D	0
$\overline{D}$	0	$\overline{D}$	X	0	$\overline{D}$

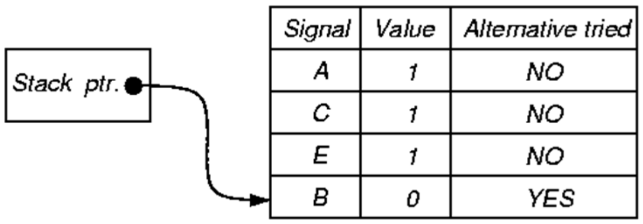
Backward Implication

- Unique determination of all gate inputs when the gate output and some of the inputs are given
- Backward implication is implemented procedurally
  - Since tables are cumbersome for gates with more than 2 inputs



Implication Stack

- Push-down stack. Records:
  - Each signal set in circuit by ATPG
  - Whether alternate signal value already tried
  - Portion of binary search tree already searched
- Example
  - PIs were set in order A, C, E, and B
  - B was set to 1 but failed

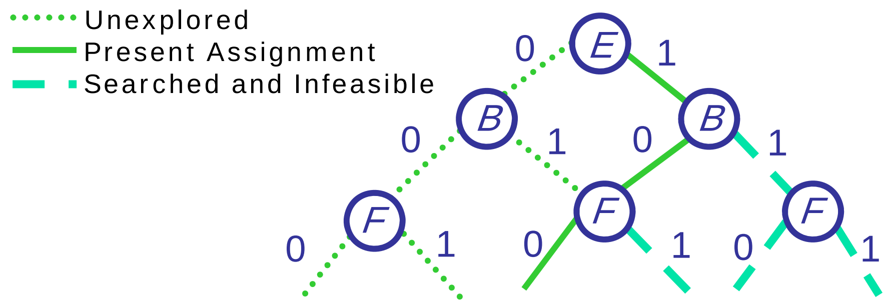


Signal	Value	Alternative tried
A	1	NO
C	1	NO
E	1	NO
B	0	YES

Implication Stack after Backtrack

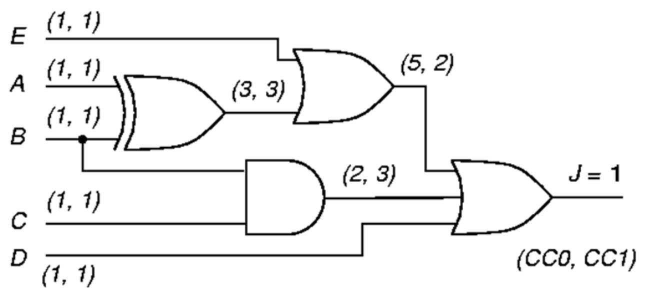
Stack ptr.

Signal	Value	Alternative tried
E	1	NO
B	0	YES
F	0	YES



Objectives and Backtracing of ATPG

- Objective: desired signal value goal for ATPG
  - Guides it away from infeasible/hard solutions
  - Intermediate signal assignments may make it impossible to achieve it
- Backtrace: Determines which primary input and value to set to achieve objective
  - Use testability measures



Branch-and-Bound Search

- An efficiently search method of binary search tree
- Branching
  - At each tree level, selects which input variable to set to what value (0 or 1)
- Bounding
  - Avoids exploring large tree portions by restricting search decision choices
  - Complete exploration is impractical
  - Decision about bounding made with limited information
    - Uses heuristics

Specific-Fault Oriented Test Generation

- Three Approaches
  - **D Algorithm**: Internal Line Values Assigned (Roth-1966)
    - D-cubes
    - Bridging faults
    - Logic gate function change faults
  - **PODEM**: Input Values Assigned (Goel – 1981)
    - X-Path-Check
    - Backtracing
  - **FAN**: Input and Internal Values Assigned (1983)