4.1 Introduction – Conceptual View

Generate an input vector that can _________ the ______ - ____ circuit from the _________ one.
4.1 Introduction –
Simple Illustration

- Consider the fault d/1 in the defective circuit
- To detect
  - _____ Fault, set d = __, means that __ = __ = 0
  - _________ Fault Effect, set c = __
- Vector: abc=_____ (output = __/____)

4.1 Introduction –
Typical ATPG System

- Given a circuit and a _____ _______
  - Repeat
    - Generate a _____ for each _________ fault
    - _____ all other faults detected by the test using a _____
  - Until ____ ______ have been considered
- Note 1: a fault may be __________, so no test can be generated
- Note 2: an ATPG may abort on a fault if the
  ____________ needed exceed a preset limit
4.2 Random Test Generation –

- _________ form of test generation
  - $N$ tests are randomly generated

- Level of confidence on random test set $T$
  - The probability that $T$ can detect ____ stuck-at faults in the given circuit
  - _______ of a random test set highly depends on the underlying circuit
  - Some circuits have many ______-______ faults

4.2 Random Test Generation – Adding Weights

- ____ input probabilities to _____ random resistant faults

- Consider an 8-input AND gate
  - Without biasing input probabilities, the probability of generating a logic 1 at the gate output = $(0.5)^8 = 0.004$
  - If we bias the inputs to 0.75, then the probability of generating a logic 1 at the gate output = $(0.75)^8 = 0.100$

- Obtaining an _________ set of input probabilities a difficult task

- Goal: increase the signal probabilities of ____-__-_____ regions
4.2 Random Test Generation – Probability of Fault Detection

Given a circuit with \( n \) inputs

Let \( T_f \) be the ___ __ _______ that can detect fault \( f \)

Then \( d_f = \frac{T_f}{2^n} \) is the probability that \( f \) can be ________ by a _____ vector

Let \( e_f = 1 - d_f \) be the probability that a random vector ______ detect \( f \)

Then, \( e_f^N = (1 - d_f)^N \) is the probability that ___ random vectors do ___ detect \( f \)

Thus, the probability that ___ one out of \( N \) random vectors can detect \( f \) is \( 1 - (1 - d_f)^N \)

4.2 Random Test Generation – Minimum Detection Probability

The ________ detection probability of any detectable fault actually does not depend on \( n \), the num of ____s

Instead, it depends on its largest ________-______

Any detectable fault must be _____ and ____________ to a primary output

Diagram: Excitation cone, Primary output cone, POs, Inputs outside of PO cone are not needed for detection of fault 

Page 7
4.2 Random Test Generation – Exhaustive Test Generation

- **Exhaustive Testing**
  - Apply ___ patterns to an ___-input combinational circuit under test (CUT).
  - All detectable faults in the combinational circuits are detected.
  - Test time maybe be _____________ ____ if the number of inputs is ______.
  - Feasible only for _____ circuits.

- **Pseudo-exhaustive Testing**
  - ______ circuit into respective cones.
  - Apply _____________ testing only to each cone.

4.3 Theoretical Background: Boolean Difference – Definition

- For a function \( f(x_1, x_2, ..., x_n) \), the Boolean Difference is defined as
  \[
  \frac{df}{dx_i} = f(x_1, x_2, ..., x_{i-1}, x_{i+1}, ..., x_n) \oplus f(x_1, x_2, ..., x_{i}, \overline{x_i}, ..., x_n)
  \]

- The Boolean Difference defines how a fault effect on \( x_i \) can be propagated to an observable output.
4.3 Theoretical Background: Boolean Difference – Test Generation

- To detect $x_i$ s-a-0, $x_i \cdot \frac{df}{dx_i} = 1$
- To detect $x_i$ s-a-1, $\overline{x_i} \cdot \frac{df}{dx_i} = 1$

Consider $y$ s-a-0

Another Fault

- Consider $w$ s-a-0
4.3 Theoretical Background: Boolean Difference – Untestable Faults

Consider \( z = a \cdot 0 \)

4.4 Designing a Stuck-at ATPG for Combinational Circuits

In general, we don’t need an _______ of vectors that can detect the target fault.

Instead, we just want to compute ___ vector quickly.

Rather than using Boolean Difference that can obtain all vectors,

- Simply use a ______-and-______ search to find one vector quickly

Deterministic ATPG has two main goals:

- ______ the target fault
- ______ the corresponding fault effect to an output
For ATPG, we want to express _____ and ____-____ behavior at the same time, use a ___

5-Value Algebra: 0(__), 1(__), X(__), D(__), D’(__).

The ATPG systematically and implicitly searches the entire search space
Whenever a ______ is found, we must go back to the _____ ______ made and make a different one, this process is called ___________.

Algorithm 2 Basic Fanout Free ATPG (C, g/v)

1: initialize circuit by setting all values to X;
2: JustifyFanoutFree(C, g, v); /* excite the fault by justifying line g to v */
3: PropagateFanoutFree(C, g); /* propagate fault effect from g to a PO */
Algorithm 3 JustifyFanoutFree(C, g, v)

1: g = v;
2: if gate type of g === primary input then
3:   return;
4: else if gate type of g === AND gate then
5:   if v = 1 then
6:     for all inputs h of g do
7:       JustifyFanoutFree(C, h, 1);
8:     end for
9:   else {v = 0}
10:   h = pick one input of g whose value === X;
11:   JustifyFanoutFree(C, h, 0);
12: end if
13: else if gate type of g === OR gate then
14:   ...
15: end if

Justify works from a ______ towards the primary inputs.

________ might be used in step 10

Consider g/0
Justify called 4 times

JustifyFanoutFree(C, __, __)  JFF(C, __, __)
JustifyFanoutFree(C, __, __)  JFF(C, __, __)
JustifyFanoutFree(C, __, __)  JFF(C, __, __)
JustifyFanoutFree(C, __, __)  JFF(C, __, __)
4.4 Designing a Stuck-at ATPG for Combinational Circuits - Justify Example #2

Consider \( g/0 \)

Then \( d/0 \)

---

**Algorithm 4 PropgateFanoutFree(\( C, g \))**

1: if \( g \) has exactly one fanout then
2: \( h = \) fanout gate of \( g \);
3: if none of the inputs of \( h \) has the value of \( X \) then
4: backtrack;
5: end if
6: if \( g \) has more than one fanout
7: \( h = \) pick one fanout gate of \( g \) that is unjustified;
8: end if
9: if gate type of \( h \) = AND gate then
10: for all inputs, \( j \), of \( h \), such that \( j \neq g \) do
11: if the value on \( j \) = \( X \) then
12: \( \) JustifyFanoutFree(\( C, j, 1 \));
13: end if
14: end for
15: else if gate type of \( h \) = OR gate then
16: for all inputs, \( j \), of \( h \), such that \( j \neq g \) do
17: if the value on \( j \) = \( X \) then
18: \( \) JustifyFanoutFree(\( C, j, 0 \));
19: end if
20: end for
21: else if gate type of \( h \) = \( \ldots \) gate then
22: \( \) \ldots
23: end if
24: \( \) PropgateFanoutFree(\( C, h \));

---
4.4 Designing a Stuck-at ATPG for Combinational Circuits - Propagate Example

Back to g/0, after justify sets g to 1, then

Consider g/0

Consider g/1
4.4 Designing a Stuck-at ATPG for Combinational Circuits - D Algorithm

• The D algorithm tries to __________ a D or D’ to a ________ by making assignments on _______ signals and primary inputs.

• _______ gates with a D or D’ on the input but not on the output

• Once D or D’ makes it to a _______ ____________, the algorithm tries to _______ the values used for __________

• _______ gates with a value set on the output but not justified by its inputs

Algorithm 5 D-Algorithm(C, f)

1: initialize all gates to don’t-cares;
2: set a fault-effect (D or D') on line with fault f and insert it to the D-frontier;
3: J-frontier = \phi;
4: result = D-Alg-Recursion(C);
5: if result == success then
6:   print out values at the primary inputs;
7: else
8:   print fault f is untestable;
9: end if
4.4 Designing a Stuck-at ATPG for Combinational Circuits - D-Alg-Recursion

1: if there is a conflict or D-frontier is $\phi$ then
2: return (failure);
3: end if
4: /* first propagate the fault-effect to a PO */
5: if no fault-effect has reached a PO then
6: while not all gates in D-frontier have been tried do
7: $G =$ an untried gate in D-frontier
8: set all unassigned inputs of $g$ to non-controlling and add to J-frontier
9: result = D-Alg-Recursion($C$);
10: if result == success then
11: return (success);
12: end if
13: end while
14: return (failure);
15: end if /*fault effect has reached at least one PO*/

16: if j-frontier is $\phi$ then
17: return (success);
18: end if
19: $g =$ a gate in J-frontier
20: while $g$ has not been justified do
21: $j =$ an unassigned input of $g$;
22: set $j = 1$ and insert $j = 1$ to J-frontier
23: result = D-Alg-Recursion($C$);
24: if result == success then
25: return (success);
26: else try the other assignment
27: set $j = 0$;
28: end if
29: end while
30: return (failure);
4.4 Designing a Stuck-at ATPG for Combinational Circuits - D-Algorithm Example #1

Target fault: f/0

4.4 Designing a Stuck-at ATPG for Combinational Circuits - D-Algorithm Example #2

Target fault: f/1
4.4 Designing a Stuck-at ATPG for Combinational Circuits - D-Algorithm Example #3

Target fault: g/1

4.4 Designing a Stuck-at ATPG for Combinational Circuits - PODEM

- In the D algorithm, the _______ space consists of ____ the lines in the circuit.
- PODEM makes _______ only at the _______ _______, eliminating any unjustified values
- Backtracks when D-frontier is ______
- Picks an objective (___________ or ___________) and traces it back to a primary input _______ making assignment
4.4 Designing a Stuck-at ATPG for Combinational Circuits - PODEM

Algorithm 9 getObjective(C)

1: if fault is not excited then
2:    return (g, ν);
3: end if
4: d = a gate in D-frontier;
5: g = an input of d whose value is x;
6: ν = non-controlling value of d;
7: return (g, ν);

Algorithm 10 backtrace(C)

1: i = g;
2: num_inversion = 0;
3: while i ≠ primary input do
4:     i = an input of i whose value is x;
5:     if i is an inverted gate type then
6:        num_inversion++;
7:     end if
8: end while
9: if num_inversion == odd then
10:    ν = \overline{ν};
11: end if
12: return(i, ν);
4.4 Designing a Stuck-at ATPG for Combinational Circuits – PODEM Example

- **1st Objective:**
- Backtrace from the objective:
- Simulate(c=0): D-Frontier =
- 2nd Objective:
- Backtrace from the objective:
- Simulate(a=0):

- **Target fault: b/0**

- 1st Objective:
- Backtrace from objective:
- Simulate(a=0):
- Must backtrack
- Change decision
- Simulate(a=1):
- Backtrack
4.4 Designing a Stuck-at ATPG for Combinational Circuits – PODEM Example

- PODEM for an improved ATPG
- The number of decision points
- Concept of
  - A ________ is the output of a ________ region, backtrace can _________ from a ________ to a primary input
- Objectives to reduce later

4.4 Designing a Stuck-at ATPG for Combinational Circuits – FAN

- PODEM for an improved ATPG
- The number of decision points
- Concept of
  - A ________ is the output of a ________ region, backtrace can _________ from a ________ to a primary input
- Objectives to reduce later
4.4 Designing a Stuck-at ATPG for Combinational Circuits – FAN

- Objectives:
- Backtrace from k=0 may favor ____, but __________ would __________ the second objective m=1!
- Choose _____ instead
- Makes backtrace more ___________ to avoid future conflicts

4.4 Designing a Stuck-at ATPG – Static Logic Implications

- _______ logic implications
- _________ logic implications
- _________ _________ implications
4.4 Designing a Stuck-at ATPG
– Static Logic Implications (Direct)

- Direct implications for \( f=1 \):
- Direct implications for \( j=0 \):

4.4 Designing a Stuck-at ATPG
– Static Logic Implications (Indirect)

- Direct implications for \( f=1 \):
- Indirect Implications for \( f=1 \) obtained by simulating the direct implications of \( f=1 \):
- This process is repeated for every node in the circuit
4.4 Designing a Stuck-at ATPG

– Static Logic Implications (Extended Backwards)

- In order to justify ____, need either ____ or ____
- Simulate(a=1, impl(f=1)) = Sa
- Simulate(b=1, impl(f=1)) = Sb
- ___________ of Sa and Sb is the the set of extended backward implications for f=1
- This process is repeated for every ________ gate, as well as for every node in the circuit

4.4 Designing a Stuck-at ATPG

– Dynamic Logic Implications

- Similar to ______ Logic Implications, but some signals ______ assigned values
- Suppose ____ has already been assigned
  - Then to obtain z=0, __________
  - d=0 requires ________, e=0 requires ________
  - The intersection of ________ and ______ is ______
4.5 Sequential ATPG – Huffman Model

To detect a fault, a sequence of vectors may be needed.
4.5 Sequential ATPG – Basic Framework

- Based on ___________ ATPG
- Targets one _________ at a time
- Excite the target fault in time-frame __ and propagate it to a __, possibly through _________ time-frames
- ______ the state needed at time-frame __, via possibly several time-frames
- Sequential ATPG very complex, as backtracks can involve reversing decisions at different time-frames

4.6 Untestable Fault Identification

- Untestable faults are:
  - Those that could not be ________, or
  - Those that could not be __________, or
  - Those that could not be ______________________ and __________
- ATPG can spend a lot of time trying to generate a test for an untestable fault
4.6 Untestable Fault Identification – FIRE Method

- Based on _____ analysis
- \( S_0 \) = set of faults that are untestable when signal _____
- \( S_1 \) = set of faults that are untestable when signal _____
- __________ of \( S_0 \) and \( S_1 \) are definitely untestable
  - They require \( s=1 \) and \( s=0 \) __________ to be detectable!

- Propagate __________
- Propagate __________

4.6 Untestable Fault Identification – FIRE Example

\[ \text{Impl}[b=1] = \]

- Faults unexcitable when \( b=1 \):
- Faults unobservable when \( b=1 \):
- Fanout stems may still be __________ even if _______ are not, due to multiple path propagation
- A fanout stem, \( s \), may be observable if both of the following are true
  - \( s \) has at least one __________ parity convergence
  - None of the uncontrollable lines involved in blocking are __________ from \( s \)
4.6 Untestable Fault Identification – FIRE Example Continued

- e satisfies the ______ but not the _______. In this case, add {________} to the unobservable list and propagate unobservability to include {_____________}

- Faults undetectable (union of unexcitable and unobservable) when b=1:

- Impl[b=0] =
- Faults unexcitable when b=0:
- Faults unobservable when b=0:
- Faults undetectable when b=0:
4.6 Untestable Fault Identification – FIRE Example Concluded

• Now that the two sets of faults undetectable when b=0 and b=1 have been computed,
  The ___________ of the two sets are those faults that require b=1 AND b=0 for detection, thus _________:
  b=1: {a/0, a/1, b/1, b/1, b/0, b/1, c/0, c/1, d/1, e/0, e/1, e/0, e/1, e/1, e/0, e/1, e/1, e/1, e/0, e/1, x/0, y/0, y/1, z/0}
  b=0: {b/0, b/0, b/0, c/0, c/1, e/0, e/0, e/0, e/1, y/1}
• Intersection and untestable:

4.6 Untestable Faults – Multiple-Line Conflict Analysis

• Consider an AND gate
• {a=0, c=1} is illegal (but this is captured by _________ conflicts)
• Likewise {b=0, c=1}
• But, {a=1, b=1, c=0} is a _________ conflict not captured by _________ conflict
• S₀ – set of faults undetectable when signal a=0
• S₁ – set of faults undetectable when signal b=0
• S₂ – set of faults undetectable when signal c=1
• Intersection of these sets is the set of undetectable faults
4.6 Untestable Faults – Multiple-Line Conflict Analysis (continued)

- Can ______ the previous concept further
- Consider multi-line conflict {_____________}
- We can extend these values as far as possible: {____ _______________} is a multi-line conflict as well

4.6 Untestable Faults – Summary

- First compute _____ logic implications
- Compute untestable faults based on __________ conflicts
- Compute untestable faults based on __________ conflicts
- __________ all identified untestable faults from the fault list
4.10 ATPG for Non-stuck-at Faults – Delay Defects

- Delay defects: class of defects that affects the __________ only when the circuit is running at __________
- __________ model insufficient to model all delay-related defects
- Delay fault models
  - ______ delay fault
  - ________ fault
  - ________ delay fault

4.10 ATPG for Non-stuck-at Faults – Delay Defects (Types of Tests)

- Launch on ________ (aka broadside or double capture)
  - V1 is arbitrary, v2 is derived from v1 through the ______ ________
- Launch on ____ (aka skewed load)
  - V1 is arbitrary, v2 is derived by a ____ shift of v1
- __________ ______
  - V1 and V2 are __________
4.10 ATPG for Non-stuck-at Faults – Delay Defects (Launch on Capture)

- True _________ test

- Benefits
  - Detect intra-clock-domain faults and inter-clock-domain _________ faults or delay faults at-speed
  - Facilitate _________ implementation
  - _________ some of functionally infeasible paths
  - Ease _________ with ATPG

4.10 ATPG for Non-stuck-at Faults – Delay Defects (Launch on Shift)

- An at-speed delay test technique
- Can address _____________ delay faults
- V1 and V2 correlated
  - ____ _____________ functionally infeasible paths
- Three approaches (details in chapter 5)
  - One-hot skewed-load
  - Aligned skewed-load
  - Staggered skewed-load
4.10 ATPG for Non-stuck-at Faults – Delay Defects (Enhanced Scan)

- Enhanced-scan cells needed
- Larger cells to hold ____ values at each FF
- Can apply two uncorrelated vectors consecutively
  - Can achieve ________ coverage, since all V1-V2 combinations are possible

4.10 ATPG for Non-stuck-at Faults – Classification of Path-Delay Faults

- Models a combinational path in the circuit
  - Considers the ______________ effect of the delay along the path
  - On-inputs of a path
  - Off-inputs of a path
- A __________ is launched at the start of the path, and a test must propagate the __________ to the end of the path
  - Two faults associated with every path: _____ and _______
  - Transition at the start of the path
- Number of paths can be ______________ to the number of gates in the circuit
- Two vectors needed
  - V1: __________ vector
  - V2: _______ and __________ vector
4.10 ATPG for Non-stuck-at Faults – Classification of Path-Delay Faults

- Statically sensitizable: all ________ of a path P can be assigned to ______________ values by some vector
- Single-path sensitizable: all ________ of a path can be set to ______________ values for both vectors of a test
- _______ path: a transition cannot propagate from the start to the end of path
  - Not all necessary off-input values can be set to non-controlling values ______________

4.10 ATPG for Non-stuck-at Faults – Path-Delay Faults (Robustly Testable)

- If a path is robustly testable, then the corresponding test can verify the correctness of the path ______________ of other _______ in the circuit
- Value criteria for robust testable path:
  - When the corresponding on-input of P has a __________ to __________ transition, the value in the first vector for the off-input can be __ with the value for the off-input as a ______________ value in the second vector.
  - When the corresponding on-input of P has a __________ to __________ transition, the values for the off-input must be a ______ non-controlling value for both vectors.
4.10 ATPG for Non-stuck-at Faults – Path-Delay Faults (Robustly Testable)

• Single-path sensitization is too ___________
• May not need to set ____ off-inputs to non-controlling values in V1 in order to propagate a transition
  – ____________ path is robustly testable

4.10 ATPG for Non-stuck-at Faults – Path-Delay Faults (Nonrobustly Testable)

• Non-robust test only valid if __ ________ delay fault is present in the circuit
• Value criteria for non-robust testing:
  – Irrespective of the ____________ on the on-input, the value in the _____ vector for the off-input can be X, with the value for the off-input being a non-controlling value in the ______ vector.
4.10 ATPG for Non-stuck-at Faults – Path-Delay Faults (Nonrobustly Testable)

- Not all paths are _________ testable
- Further ______ requirements for V1
- Test is valid if circuit has __ ____ delay faults
  - Highlighted path is _______________ testable

4.10 ATPG for Non-stuck-at Faults – Path-Delay Faults (Symbols)

- Can use new _______ to consider _____ _______ simultaneously during ATPG
- S0 – Initial and final values are both logic 0
- S1- Initial and final values are both logic 1
- U0 – Initial logic can be either 0 or 1, but final value is logic 0
- U1 – Initial logic can be either 0 or 1, but final value is logic 0
- XX - Both initial and final values are don’t cares
### 4.10 ATPG for Non-stuck-at Faults – Path-Delay Faults (Boolean Operations)

<table>
<thead>
<tr>
<th>AND</th>
<th>S₀</th>
<th>U₀</th>
<th>S₁</th>
<th>U₁</th>
<th>XX</th>
<th>NOT</th>
</tr>
</thead>
<tbody>
<tr>
<td>S₀</td>
<td>S₀</td>
<td>S₀</td>
<td>S₀</td>
<td>S₀</td>
<td>SO</td>
<td>S₁</td>
</tr>
<tr>
<td>U₀</td>
<td>S₀</td>
<td>U₀</td>
<td>U₀</td>
<td>U₀</td>
<td>U₀</td>
<td>U₁</td>
</tr>
<tr>
<td>S₁</td>
<td>S₀</td>
<td>U₀</td>
<td>S₁</td>
<td>U₁</td>
<td>XX</td>
<td>S₁</td>
</tr>
<tr>
<td>U₁</td>
<td>S₀</td>
<td>U₀</td>
<td>U₁</td>
<td>U₁</td>
<td>XX</td>
<td>U₁</td>
</tr>
<tr>
<td>XX</td>
<td>S₀</td>
<td>U₀</td>
<td>XX</td>
<td>XX</td>
<td>XX</td>
<td>XX</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OR</th>
<th>S₀</th>
<th>U₀</th>
<th>S₁</th>
<th>U₁</th>
<th>XX</th>
</tr>
</thead>
<tbody>
<tr>
<td>S₀</td>
<td>S₀</td>
<td>U₀</td>
<td>S₁</td>
<td>U₁</td>
<td>XX</td>
</tr>
<tr>
<td>U₀</td>
<td>U₀</td>
<td>U₀</td>
<td>S₁</td>
<td>U₁</td>
<td>XX</td>
</tr>
<tr>
<td>S₁</td>
<td>S₁</td>
<td>S₁</td>
<td>S₁</td>
<td>S₁</td>
<td>S₁</td>
</tr>
<tr>
<td>U₁</td>
<td>U₁</td>
<td>U₁</td>
<td>S₁</td>
<td>U₁</td>
<td>U₁</td>
</tr>
<tr>
<td>XX</td>
<td>XX</td>
<td>XX</td>
<td>S₁</td>
<td>U₁</td>
<td>XX</td>
</tr>
</tbody>
</table>

- Assumes a ________ delay is present at a circuit node
- ________ of which path the effect is propagated, the gross delay defect will be late arriving at an __________ point
- ________ used in industry
  - ________ and number of faults ________ to circuit size
  - Also needs 2 vectors to test
- Node x slow-to-rise (x-STR) can be modeled simply as two stuck-at faults
  - First time-frame: ___ needs to be ________
  - Second time-frame: ___ needs to be ________ and ________
4.10 ATPG for Non-stuck-at Faults – Path-Delay Faults (Transition Fault Model ATPG)

- Simply treat each transition fault as two stuck-at faults
- Can test it with ________, __________, or __________ ____
- __________ ____
  - First perform ATPG for ________ faults
  - Then build a __________ for the ________ generated
  - Use the __________ to identify __________ for each transition fault

4.10 ATPG for Non-stuck-at Faults – Path-Delay Faults (Transition Test Chains)

<table>
<thead>
<tr>
<th>Vectors</th>
<th>Excited Faults</th>
<th>Detected Faults</th>
</tr>
</thead>
<tbody>
<tr>
<td>V₁</td>
<td>a/0, b/1, c/1, d/0, e/0</td>
<td>a/0, b/1</td>
</tr>
<tr>
<td>V₂</td>
<td>b/1, c/0, d/0, e/1</td>
<td>c/0, d/0, e/1</td>
</tr>
<tr>
<td>V₃</td>
<td>a/1, c/1</td>
<td>a/1, c/1</td>
</tr>
<tr>
<td>V₄</td>
<td>a/1, b/0, d/1, e/0</td>
<td>b/0, d/1, e/0</td>
</tr>
</tbody>
</table>
4.10 ATPG for Non-stuck-at Faults – Path-Delay Faults (Bridging Fault ATPG)

- Models ______ between two circuit nodes
- The bridge fault is not excited unless the two circuit nodes have ________ logic values
- Faulty value depends on the bridge-fault type:
  - ____ bridge: faulty value is the _____ of the two involved nodes’ values
  - ____ bridge: faulty value is the ___ of the two involved nodes’ values
  - X Dom y: value of x __________
  - X Dom1 y: x __________ y if ____
  - X Dom0 y: x __________ y if ____

• Testing for bridging faults is similar to a _____ stuck-at ATPG.

Consider AND-bridge(x,y), we can do either:
- Detect x/0 with setting y=0
- Detect y/0 with setting x=0

_____ stuck-at ATPG can be modified to handle _____ faults
4.11 Other Topics in Test Generation – Test Compaction

- Want to reduce the _____ _____ ______
to reduce test data _____ and test
___________ time
- First build a detection _______
  - _____ vector: a vector that
detects some faults that no other vector
can detect
  - __________
- A set _______ algorithm is applied
to find a _________ test set such that
every fault is covered
- If vectors are ___________ specified
  - Some vectors may be ________:______
  - Just one vector ______ is sufficient

<table>
<thead>
<tr>
<th></th>
<th>$f_1$</th>
<th>$f_2$</th>
<th>$f_3$</th>
<th>$f_4$</th>
<th>$f_5$</th>
<th>$f_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_1$</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$r_2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$r_3$</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>$r_4$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4.11 Other Topics in Test Generation – N-Detect ATPG

- Idea: detect every fault at ______ ___ times
  - N vectors that detect a fault must be __________
- Although the same _____ coverage, can significantly enhance the _____ coverage
  - If x/0 is detected 2 times, one with ____ , and the other with ____ , then the ____-bridge fault of (x,y)
    would have been detected by the ____ test
- ATPG can be modified to N-Detect ATPG