ECE 474A/57A
Computer-Aided Logic Design

Lecture 16
Automatic Test Generation for Combinational Circuits

Introduction
Verification vs Test

- Verification
  - Predictive analysis to ensure that the synthesized design will perform the given functionality
- Test
  - Manufacturing step that ensures that the physical device has no manufacturing defect

"Program testing can be used to show the presence of bugs, but never to show their absence!"

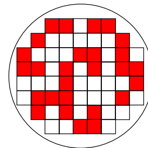
(Edsger Dijkstra)

Introduction
Terminology

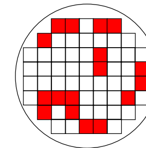
- Yield (Y) of a manufacturing process is the fraction of fault-free products
- Defect level (DL) indicates fraction of defective parts after testing
- Coverage (T) indicates fraction of fault that can be detected given a testing scheme
  - 1 = perfect test
  - 0 = completely ineffective test

$DL = 1 - Y^{(1-T)}$

DL : defect level
Y : yield
T : fault coverage of test



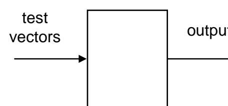
27 bad, 33 good
DL = 27/60 = 45%
Yield = 33/60 = 55%



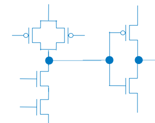
17 bad, 43 good
DL = 17/60 = 28%
Yield = 43/60 = 72%

Introduction

- Testing Methods
  - Functional Testing
  - Structural Testing
  - Others too …
- Functional Testing
  - No assumptions about circuit under test (CUT)
  - Want to verify CUT performs as expected
- Structural Testing
  - Based on assumed fault set
    - Fault is an alteration to the structure of a fault-free circuit
  - Want to verify if any of the faults in our set present on the CUT



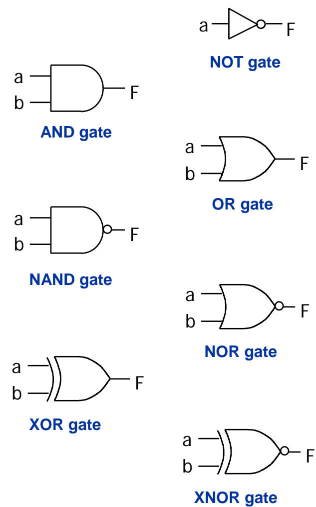
Does the behavior match?



Does the structure match?

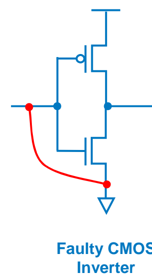
Faults and Fault Models

- Consider structural testing of combinational circuits composed of simple gates
  - Concepts can be applied to sequential testing too
- Where does fault come from?
  - Shorts, defective soldering, etc.
- Abstract to logic-level and consider effect fault has on circuit



Faults and Fault Models

- Short between input lead and ground
  - Causes the input to be stuck at 0
  - At logic level, represented as stuck-at-0 fault
- Many fault models available
  - Stuck-at faults
    - Single stuck-at-faults
    - Multiple stuck-at-faults
  - Stuck-open faults
  - Bridging Faults
  - Delay Faults

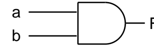


Faulty CMOS Inverter

Faults and Fault Models

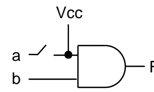
- Single stuck-at-0/1 Fault Model
  - Only one line in the circuit is faulty at a time
  - The effect of the fault is if the node is tied to Vcc (s-a-1) or Gnd (s-a-0)
  - The fault is permanent (not transient)
  - The function of the gate is unaffected by the fault
- Advantages
  - Reduced number of faults to consider
  - Well studied algorithms for automatic test pattern generation (ATPG)
  - Model cover ~90% of manufacture faults
- Disadvantages
  - Does not cover all defects

Fault-free AND Gate Behavior



a	b	F
0	0	0
0	1	0
1	0	0
1	1	1

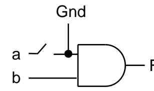
Faulty Behavior



s-a-1 fault

a	b	F
0	0	0
0	1	1
1	0	0
1	1	1

Faulty Behavior



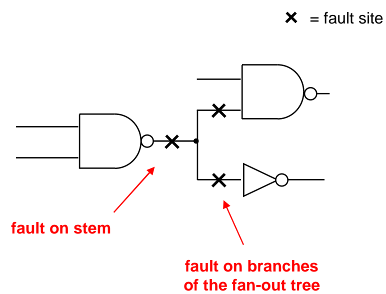
s-a-0 fault

a	b	F
0	0	0
0	1	0
1	0	0
1	1	0

Faults and Fault Models

- More than 2 faults can be defined for a single wire if wire drives multiple gates
- Total number of faults is 2N where N is the number of gate terminals
  - We have opportunity to reduce this

✕ = fault site

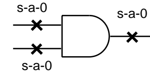


fault on stem

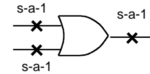
fault on branches of the fan-out tree

Faults and Fault Models

- If n faults are equivalent, it sufficient to generate a test for one of the faults
- Equivalent faults
  - All stuck-at-0 faults equivalent in 2-input AND gate
  - All stuck-at-1 faults equivalent in 2-input OR gate
  - If no fanout, stuck-at-1 and stuck-at-0 faults equivalent to gates terminals connected by wire



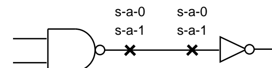
all stuck-at-0 faults at the terminal of an AND gate are equivalent



all stuck-at-1 faults at the terminal of an OR gate are equivalent

Definition 12.2.1

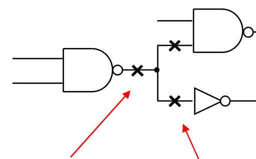
Let f_1 and f_2 be two faults of a circuit C. Let F be the function performed by C when no fault is present. Let F_1 and F_2 be the functions performed by C in the presence of f_1 and f_2, respectively. Then faults f_1 and f_2 are equivalent if and only if $F_1 = F_2$



no fanout -- faults at the output of NAND gate are equivalent to faults at the input of INV

Faults and Fault Models

- Fault Collapsing
  - Processes of identifying equivalent faults
  - Can be difficult, we assume this is done prior to test generation
- Untestable (Undetectable faults)
  - Possible that a fault f_1 does not alter behavior of circuit ($F_1 = F$)
  - Sometimes called redundant because always associated with redundancy in circuit
- Multiple stuck-at-faults possible too
  - Simultaneous presence of single stuck-at faults present in circuit
  - We ignore due to complexity

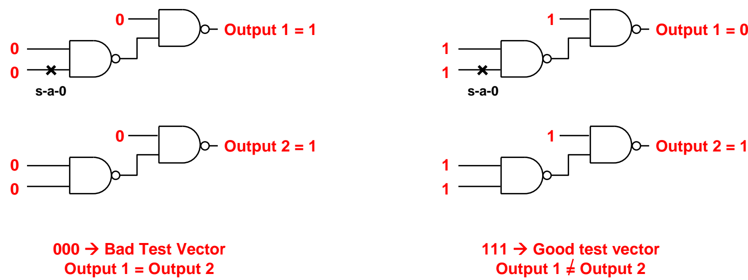


fault on stem ≠ fault on branches of the fan-out tree

In general, faults on the stem are not equivalent to faults on the branches

Automatic Test Generation

- Want to develop a procedure to generate a test for a given stuck-at fault
  - Test for a combination circuit is an assignment of 0's and 1's to the primary inputs of the circuit that causes different output in the good and fault circuits



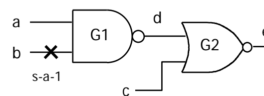
000 → Bad Test Vector
Output 1 = Output 2

111 → Good test vector
Output 1 ≠ Output 2

Automatic Test Generation
Excitation and Sensitization

- How to test for a stuck-at-1 fault on input b?

Only test for this fault is abc = 101



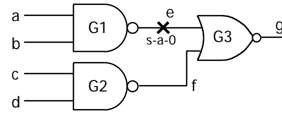
Value of b?	Value of a?	Value of c?
b = 0 must be in the test, otherwise output of G1 not dependent on presence of the fault	a = 0 G1 output always 1 regardless of other inputs, no fault detection a must be set to 1	c = 1 G1 output always 0 regardless of other inputs, no fault detection c must be set to 0

excitation of the fault - test must cause value complementary to faulty value to appear at the fault site

sensitized path – connects the fault site to the output of the circuit

Automatic Test Generation
Excitation and Sensitization

- How to test for a stuck-at-0 fault on e?
- Excite the fault
  - e must be 1 in a fault-free circuit
  - how to get e=1 using primary inputs
    - a=0, b=0
    - a=0, b=1
    - a=1, b=0
- Propagate the fault to g
  - Sensitization of a path to output requires f=0
    - c=1, d=1

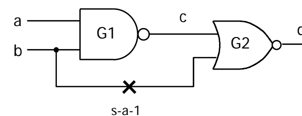


Possible test vectors to detect fault
- **abcd = 0011**
- **abcd = 0111**
- **abcd = 1011**

Several test vectors available, usually we only need one

Automatic Test Generation
Excitation and Sensitization

- How to test for a stuck-at-1 fault on input of G2 driven by b?
  - Excite the fault
    - b must be equal to 0
  - Propagate the fault to g
    - Sensitization of a path to output requires c=0
      - a=1, b=1
- Reconvergent fanout
  - Paths that have a common source (e.g. b) and common sink (e.g. G2)
  - Poses problems with test generation



Requirements are contradictory

Not always possible to find test vectors to detect fault

Cannot test for s-a-1 on input to G2

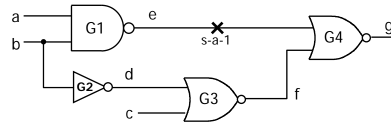
Cannot test for s-a-1 on stem of b

CAN test for s-a-1 on the input of G1 connected to b

Automatic Test Generation

Excitation and Sensitization

- Not all conflicts indicate fault untestable
  - Excitation and sensitization condition may be satisfied in different ways
  - If an arbitrary decision leads to conflict later, return to decision point and try an alternative
- Backtracking
  - Returning on one's step and reversing a previous choice
  - Larger circuits difficult to avoid "wrong" choices that need to be reversed
    - Ability of test generation algorithms lies in guessing the correct choice and reducing amount of backtracking required



Test vectors to detect s-a-1 fault at e

Excitation Condition
- **a = 1, b = 1**

Sensitization of Path
- **f = 0**
  - **try d = 1, c = 1** — **Won't work, to get d = 1 we need b = 0**
  - **try d = 0, c = 1** — **Works!**

Solution: abc= 111

Automatic Test Generation

Excitation and Sensitization

- Controlling Value
  - A value when present on at least one input forces an output to a known value (controlled value)
- Non-controlling Value
  - Complement of controlling value
  - Used to sensitize a path
    - Set side inputs to non-controlling value

AND gate

a	b	F
0	0	0
0	1	0
1	0	0
1	1	1

Controlling value = 0
Controlled value = 0

NAND gate

a	b	F
0	0	1
0	1	1
1	0	1
1	1	0

Controlling value = 0
Controlled value = 1

OR gate

a	b	F
0	0	0
0	1	1
1	0	1
1	1	1

Controlling value = 1
Controlled value = 1

NOR gate

a	b	F
0	0	1
0	1	0
1	0	0
1	1	0

Controlling value = 1
Controlled value = 0

XOR gate

a	b	F
0	0	0
0	1	1
1	0	1
1	1	0

Controlling value = None
Controlled value = None

XNOR gate

a	b	F
0	0	1
0	1	0
1	0	0
1	1	1

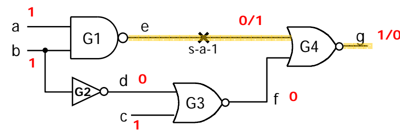
Controlling value = None
Controlled value = None

For XOR and XNOR gate we must realize an odd number of inputs should be sensitized if we want output to be sensitized

Automatic Test Generation

A Simple Test Generation Algorithm

- Key idea for testing
  - Generate test so that difference from good and faulty circuit
  - Propagated this difference to a primary output by creating sensitized path(s)
- Values of good and faulty circuit are complementary along sensitized path
  - Indicated by compound values 1/0 or 0/1
  - First value is fault-free value
- Symbol D (defect) used to represent these compound values
  - D represents 1/0
  - $\overline{D}$ represent 0/1

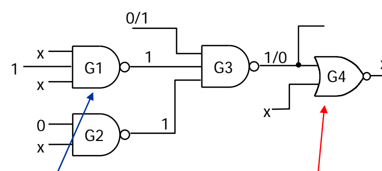


NOTE:
0/0 values indicated by 0
1/1 values indicated by 1

Automatic Test Generation

A Simple Test Generation Algorithm

- Simple algorithm for test generation
  - Inputs - description of the circuit and a single stuck-at fault
  - Output - test or indication that fault is untestable
- Basic Idea
  - Initializes all lines to unassigned (x)
  - Builds test by assigning values (1, 0, 1/0, 0/1) to lines required for excitation and sensitization



Unjustified Element

Output assigned, but assigned inputs do not imply output specified

Helps to decide when we terminate algorithm

Frontier Element

Output unassigned, one of its input carries a compound value

Indicates how far symptoms of the fault have been propagated

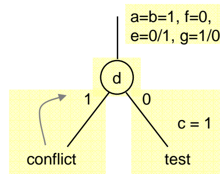
Q: Is G2 an unjustified element?

A: No. 0 is a controlling value that ensures output of G2 is 1

Automatic Test Generation

A Simple Test Generation Algorithm

- Decision tree
  - Track multiple, cascaded choices
  - Ensure orderly examination of solution space
- Nodes correspond to signals for which choice is made
  - Annotate signals whose values are implied by choice made along arc
- Objective achieved when
  - Frontier propagated to one of the primary outputs
  - No unjustified lines are left



values uniquely determined by the excitation and sensitization conditions

If we decide to set d=1, we propagate the implications of this choice and find conflict with b

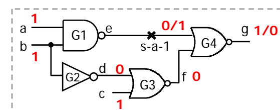
We face a choice – find input assignment to make f=0

make node d, make choice

Abandon path, backtrack to last decision

set d = 0, forces c to 1

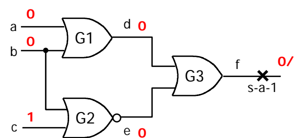
OBJECTIVE ACHIEVED!



Automatic Test Generation

Implications and Backtracking

- Simplify each node of the decision tree by finding as many possible implications of the choices made along the path



excitation condition?

f = 0 in fault free circuit

f=0 implies d=0, e=0

d = 0 implies a=0, b=0

e = 0, b=0 implies c=1

f=0
d=0, e=0
a=0, b=0
c = 1

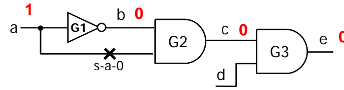
test

Just by propagating implications we find a test vector

Automatic Test Generation

Implications and Backtracking

- What happens when we propagate the implications of the excitation condition in this example?



excitation condition?

a = 1

a=1 implies b=0, c=0

c = 0 implies e=0

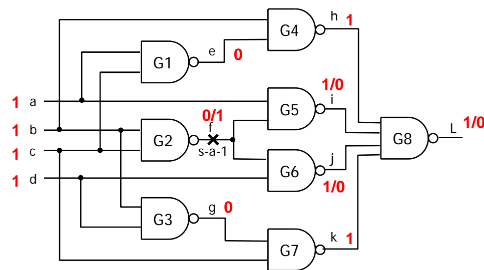
no frontier elements which means no test (e set to 0 regardless of d)

No test for s-a-0 fault above

Automatic Test Generation

Implications and Backtracking

- How does propagation of implications help in this example?



Excitation condition?

f = 0 in fault-free circuit

f = 0 implies b=1, c= 1

Values has to come out of G8

Paths not connected to the fault need to be sensitized

h=1, k=1

Any implications?

if b=1 and want h=1, implies e=0

if c=1 and want k=1, implies g=0

if c=1 and want e=0, implies a=1

if b=1 and want g=0, implies d=1

given a=1 and f=0/1, i is implied to be 1/0

given d=1 and f=0/1, i is implied to be 1/0

given h=1, i=1/0, j=0/1, k=1, L is implied to be 1/0

Automatic Test Generation

Implications and Backtracking

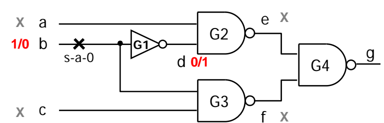
- Outline of test generation algorithm

1. Apply the fault excitation condition
2. Perform the implications of the last assignment
3. If the fault symptoms have reached at least one primary output, justify the remaining unjustified lines. If the justification fails, backtrack and go to Step 2. Otherwise, exit: a test has been found
4. If the frontier is empty, backtrack and go to Step 2
5. If the frontier consists of one gate only, perform the resulting implications (this is discussed later) and go to Step 2.
6. Choose one signal that is not reachable form the fault site and assign to it either 1 or 0. Create a corresponding node in the decision tree. Go to Step 2.

Automatic Test Generation

Implications and Backtracking

- Let's try out the ATPG, all lines initially set to X



(Step 1) Fault excitation condition

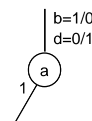
b =1 in fault free circuit, b = 0 if s-a-0 exists

(Step 2) Implications of the last assignment

d = 0/1

(Step 3) Fault symptoms have reached at least one primary output?

No primary output reached



(Step 4) Frontier empty?

No. Frontier consists of G2 and G3

(Step 5) Frontier consists of one gate only?

No. Frontier consists of G2 and G3

(Step 6) Choose one signal that is not reachable form the fault site and assign to it either 1 or 0. Create a corresponding node in the decision tree. Go to Step 2.

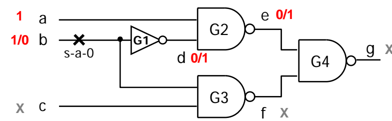
a = 1

Goal in selecting a = 1 is to move frontier forward to G4 by allowing propagation of fault symptoms through G2

Automatic Test Generation

Implications and Backtracking

- Let's try out the ATPG, all lines initially set to X



(Step 2) Implications of the last assignment

if a = 1, then e = 0/1

(Step 3) Fault symptoms have reached at least one primary output?

No primary output reached

(Step 4) Frontier empty?

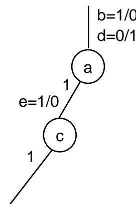
No. Frontier consists of G3 and G4

(Step 5) Frontier consists of one gate only?

No. Frontier consists of G3 and G4

(Step 6) Choose one signal that is not reachable form the fault site and assign to it either 1 or 0. Create a corresponding node in the decision tree. Go to Step 2.

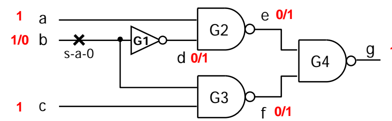
c = 1



Automatic Test Generation

Implications and Backtracking

- Let's try out the ATPG, all lines initially set to X



(Step 2) Implications of the last assignment

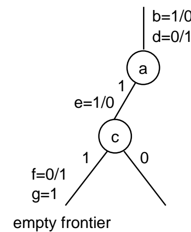
if c = 1, then f = 1/0 and g = 1

(Step 3) Fault symptoms have reached at least one primary output?

No primary output reached

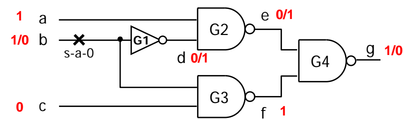
(Step 4) Frontier empty?

Yes. Backtrack to step 2 and reverse the last choice.



Automatic Test Generation
Implications and Backtracking

- Let's try out the ATPG, all lines initially set to X



(Step 2) Implications of the last assignment

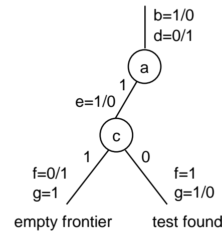
if c = 0, then f = 1 and g = 1/0

(Step 3) Fault symptoms have reached at least one primary output?

Yes!

No unjustified lines remain.

A test has been generated (110)



Automatic Test Generation
Choice of Decision Variables

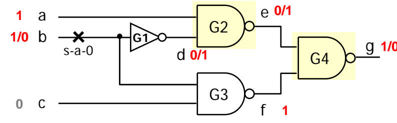
- Outline of test generation algorithm

1. Apply the fault excitation condition
2. Perform the implications of the last assignment
3. If the fault symptoms have reached at least one primary output, justify the remaining unjustified lines. If the justification fails, backtrack and go to Step 2. Otherwise, exit: a test has been found
4. If the frontier is empty, backtrack and go to Step 2
5. If the frontier consists of one gate only, perform the resulting implications (this is discussed later) and go to Step 2.
6. Choose one signal that is not reachable form the fault site and assign to it either 1 or 0. Create a corresponding node in the decision tree. Go to Step 2.

How to choose?

Automatic Test Generation
Choice of Decision Variables

- How to choose?
  - Only restriction is the line we choose is not reachable from fault site
  - Can choose primary inputs or internal lines
  - Can choose unjustified and unassigned elements
- Method 1 (Original D-algorithm)
  - Always select an unassigned input to a frontier gate
  - Keep going until primary output reached or frontier disappears
  - Then select unassigned input to unjustified gate



Select an unassigned input to a frontier gate

G2 is a frontier gate

set a = 1

Keep going until primary output reached or frontier disappears

G4 is a frontier gate

set f = 1

Primary output reached, main ATPG algorithm will assigned input to unjustified gate (step 3)

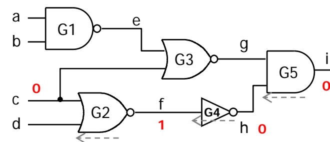
G3 is a unjustified gate

set c = 0

Automatic Test Generation
Choice of Decision Variables

- Method 2 (PODEM Algorithm)
  - Always select a primary input
    - Uses sub-procedure called backtrace to choose
  - Backtrace()
    - Given objective - line an desired value
    - Traces backward path until input found
  - Result of Backtrace() is heuristic!
    - c= 0 does not guarantee i=0
    - Choice of which input to follow heuristic
- Method 3 (FAN Algorithm)
  - Select fanout points or head-lines
    - Head-line is line where all preceding gates do not fanout
  - Similar mechanism to backtrace used
  - Idea – reconvergent fanout caused problems, choose fanout assignments in hopes of exposing conflict early

We'll rely on intuition (i.e. pick one)



Backtrace called – initial objective set i = 0

Goes (backwards) through G5

Chooses one of the inputs as having 0 as objective

Objective set h = 0

Goes through G4, f needs to be 1

Objective set f = 1

Goes (backwards) through G2

Chooses one of the inputs as having 0 as objective

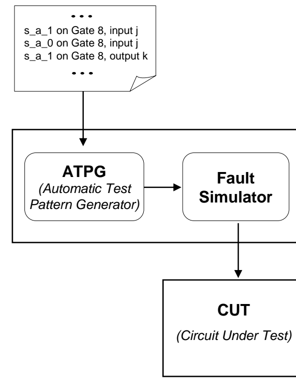
Objective set c = 0

Primary input reached, procedure terminates

Automatic Test Generation

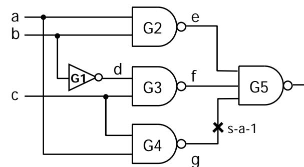
Putting the Pieces Together

- Have algorithm to generate test for a given fault, now what?
  - Generate all possible stuck-at-faults
  - Identify equivalent faults
    - Choose one – representative fault
  - Set of representative faults passed to ATPG program
- ATPG program/Fault simulator work together
  - Fault simulator is program that determines which faults from given list are detected
  - ATPG picks on fault from list
    - Generate test and pass to fault simulator for check
    - Can't test, mark untestable and remove from list



Redundancy Removal

- ATPG algorithm can also be used to simplify circuits – *Redundancy Removal*
- An untestable stuck-at-fault signals redundancy in the circuit
  - Stuck-at-1 fault untestable, can be replaced with a constant 1 without changing functionality of circuit



Try to generate test for g s-a-1 fault

a = c = 1 need to excite fault (g = 0)

e = f = 1 to sensitize

to get e = 1, a = 1 implies b = 0

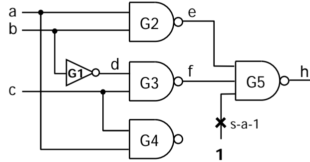
to get f = 1, c = 1 implies b = 1

g s-a-1 fault untestable

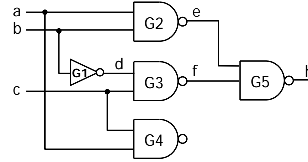
Observation says that s-a-1 can be set to constant 1

Redundancy Removal

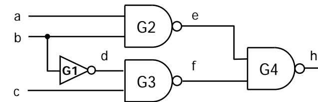
- Example Continued



Observation says that s-a-1 can be set to constant 1



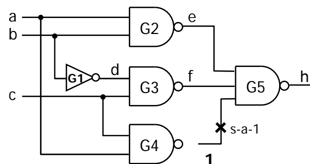
Remove input from G5 (non-controlling value)

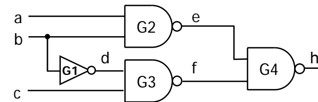


Remove G4, not connected to anything

Redundancy Removal

- Did it work?
  - Verify with truth table, simulation, ...





Redundancy Removal

- General simplifications
  - Input line of gate is fixed to controlling value
    - Replace gate with constant controlled value
  - Input line of gate is fixed to non-controlling value
    - Input to gate removed

NAND gate

a	b	F
0	0	1
0	1	1
1	0	1
1	1	0

a	b	c	F
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Controlling value = 0
Controlled value = 1

input fixed to controlling value
replace with controlled value of gate

input fixed to non-controlling value
remove gate input

2-input gate special case
one input fixed to non-controlling value, turns into inverter

Redundancy Removal

- Similar observations for other gates
  - AND, OR, NOR
- Propagate constant values
  - If pair of cascaded inverters found, remove

AND gate

a	b	F
0	0	0
0	1	0
1	0	0
1	1	1

Controlling value = 0
Controlled value = 0

NAND gate

a	b	F
0	0	1
0	1	1
1	0	1
1	1	0

Controlling value = 0
Controlled value = 1

OR gate

a	b	F
0	0	0
0	1	1
1	0	1
1	1	1

Controlling value = 1
Controlled value = 1

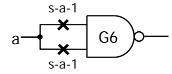
NOR gate

a	b	F
0	0	1
0	1	0
1	0	0
1	1	0

Controlling value = 1
Controlled value = 0

Redundancy Removal

- Multiple redundancies cannot be removed simultaneously
- There are exceptions, but most redundancy removal is serial

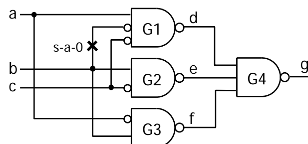


Both input s-a-1 faults untestable

Cannot replace both with constant values

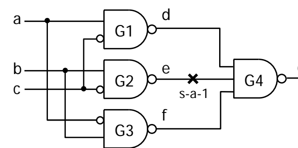
Redundancy Removal

- Let's try an example



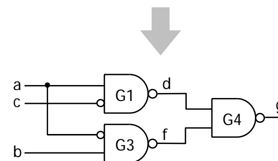
We're determined s-a-0 fault is untestable, set line to 0

Input to G1 is 1 (inverted input), a non-controlling constant means we can remove input



After simplification, we determine s-a-1 fault untestable, set line to 1

Input to G4 is a non-controlling constant means we can remove input



Simplified circuit

Redundancy Removal

- Result will differ based on order in which redundancies removed
- Benefit of redundancy removal
  - Increases testability
  - Reduces area
  - Improved performance (usually)
- Counterexample
  - s-a-0 at control input of mux in a carry-skip adder
  - Removal of redundancy transforms circuit into ripple-carry adder
    - Slower circuit
  - Specialized techniques used to address

2-bit carry-skip adder

Conclusions

- Faults and Fault Models
- Automatic Test Generation
  - Excitation and Sensitization
  - Simple Test Generation Algorithm
  - Implications and Backtracking
  - Choice of Decision Variable
- Redundancy Removal

- Whole courses on verification and testing
- Many other related topics...
  - Fault Dominance, Checkpoints, Multiple Stuck-at-Faults, Test for Design, and more.