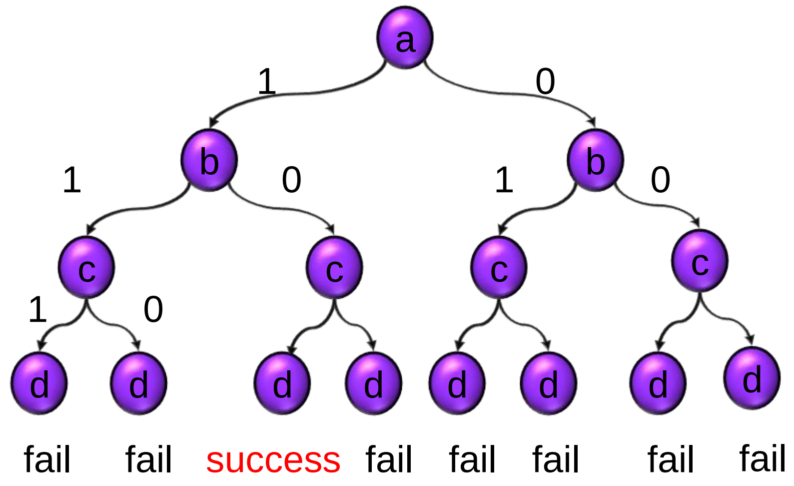
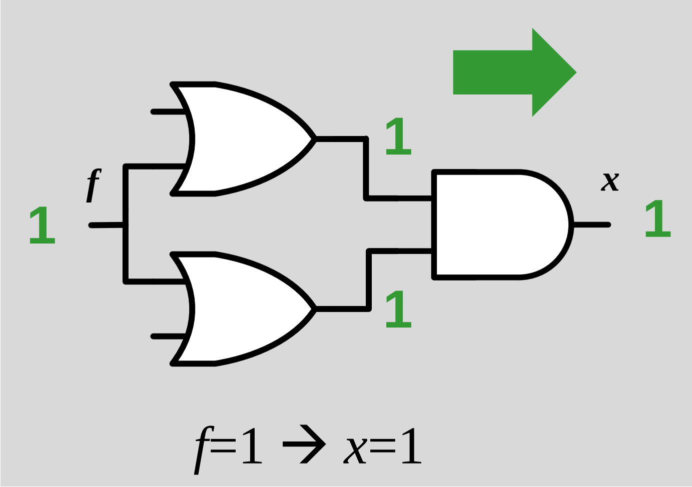
Combinational ATPG

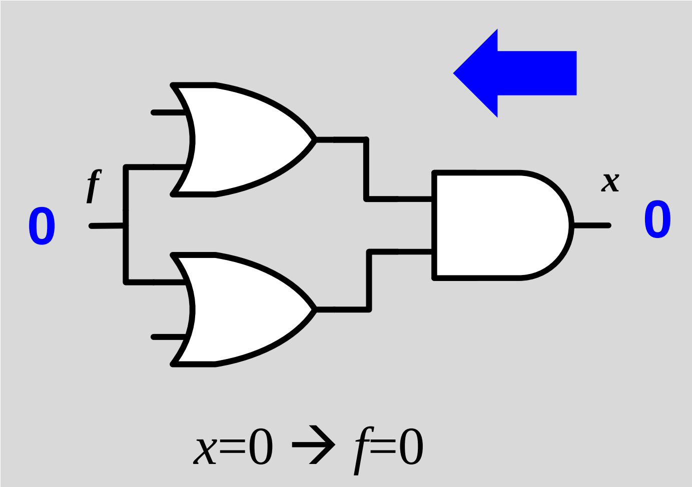
- Introduction
- Deterministic Test Pattern Generation
- <u>Acceleration techniques</u>
  - <u>Learning [Schulz 1988]</u>
  - Redundant fault identification [Iyer 1996]
- Concluding Remarks



Learning

- *Learning* memorizes circuit information to speed up test generation
  - *Static learning:* performed in preprocess. no test pattern required.
  - *Dynamic learning*: performed in test generation (not in lecture)
- Static learning example: WWW Fig 4.22
  - Set *f=1*, implies *x=1*
  - So we learn *x=0*, implies *f=0*



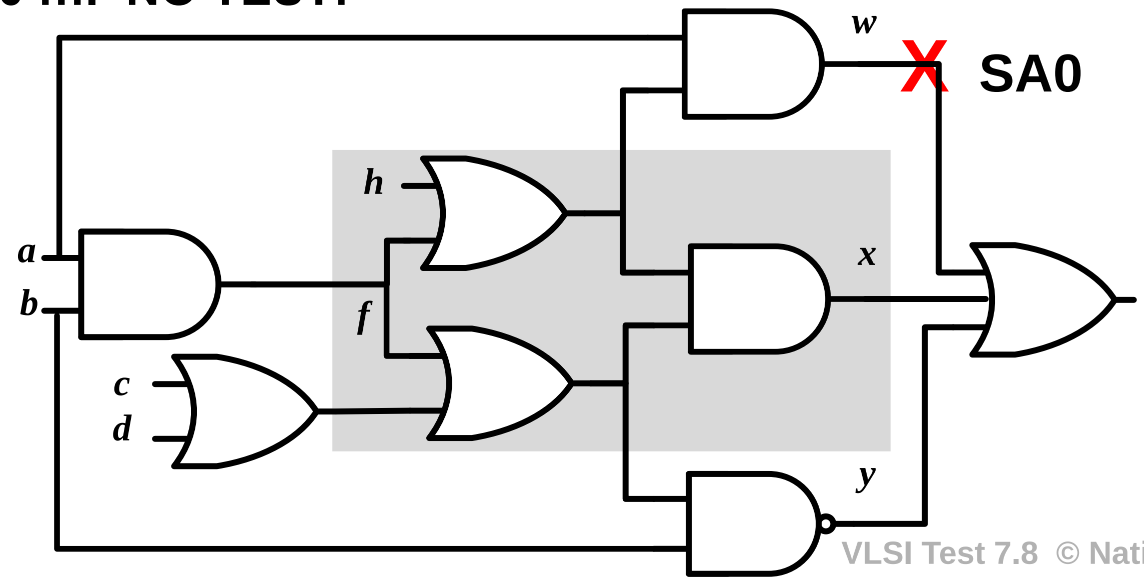


Contrapositive Law: *If p→q then q'→p'*

Learning Reduces Backtracks (1)

without learning

- Obj : w=1. assign h=1, a=1
- Obj : x=0. assign c=0, d=0, b=0
  - y=1 conflict!
- backtrack b=1
  - x=1 conflict!
- backtrack d=1, conflict!
- backtrack c=1, conflict!
- backtrack a=0, conflict!
- backtrack h=0 …. NO TEST!



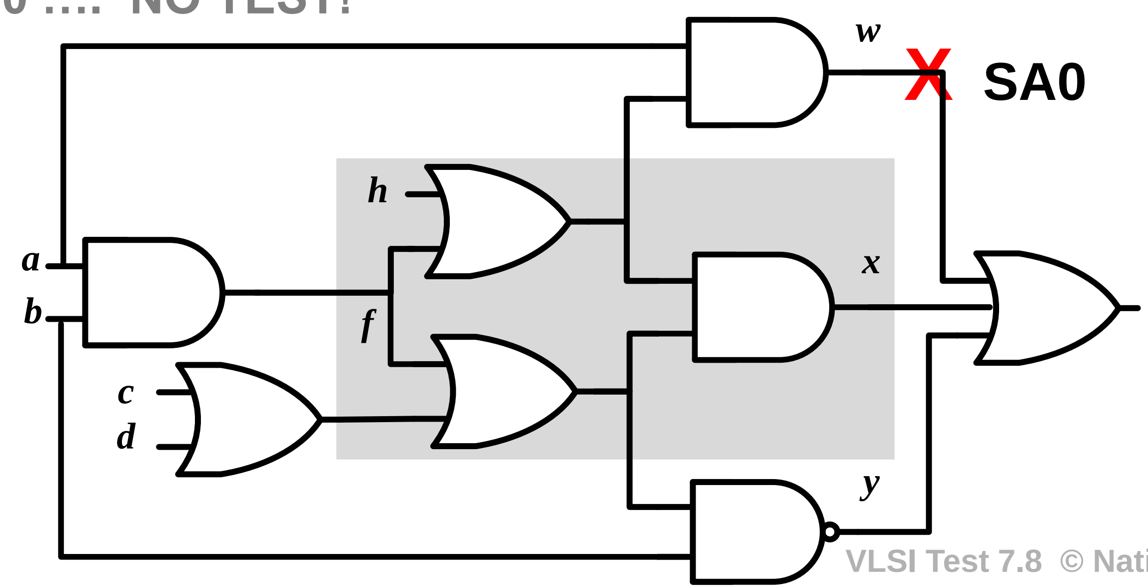
Learning Reduces Backtracks (2)

without learning

- Obj : w=1. assign h=1, a=1
- Obj : x=0. assign c=0, d=0, b=0
  - y=1 conflict!
- backtrack b=1
  - x=1 conflict!
- backtrack d=1, conflict!
- backtrack c=1, conflict!
- backtrack a=0, conflict!
- backtrack h=0 …. NO TEST!

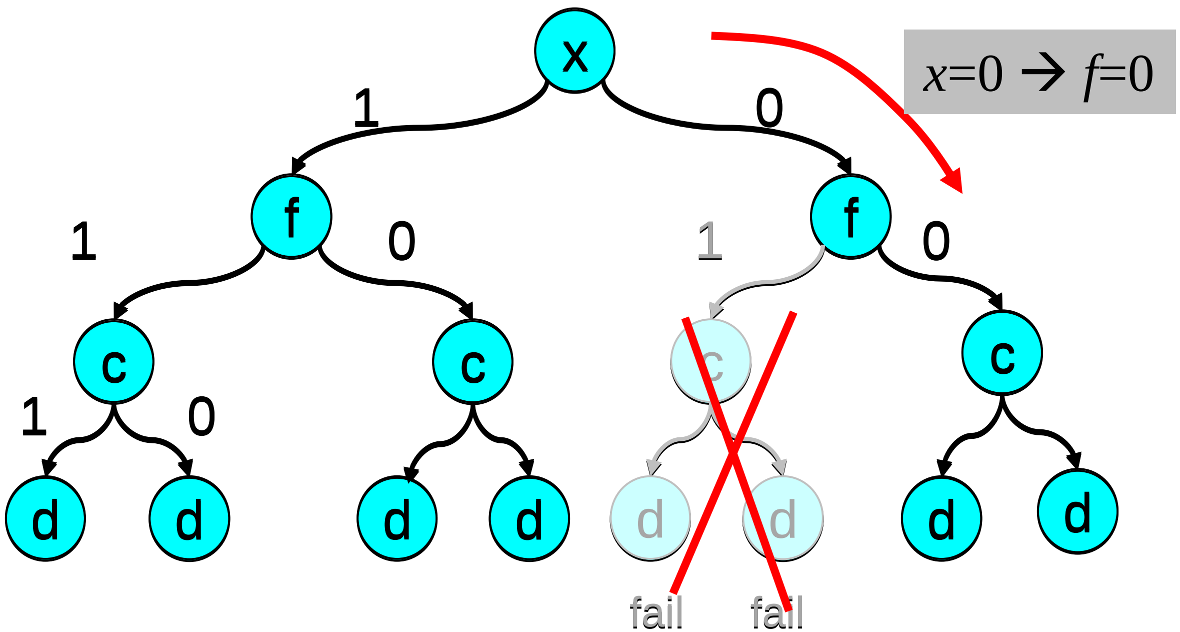
with learning

- Obj : w=1. assign h=1, a=1
- Obj : x=0 → f=0. assign b=0
  - y=1 conflict!
- backtrack b=1
  - x=1 conflict!
- backtrack a=0, conflict!
- backtrack h=0 … NO TEST!



Learning Speedup Decision

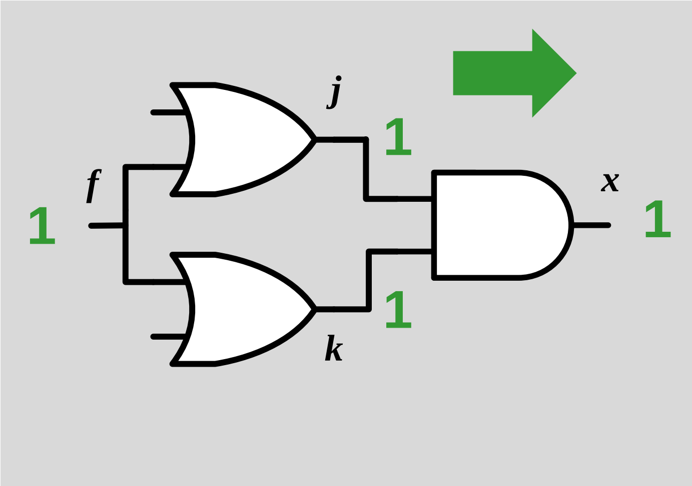
- **Learning helps to**
  - **1. Prune impossible sub-trees ASAP**
  - **2. Find necessary assignments ASAP**

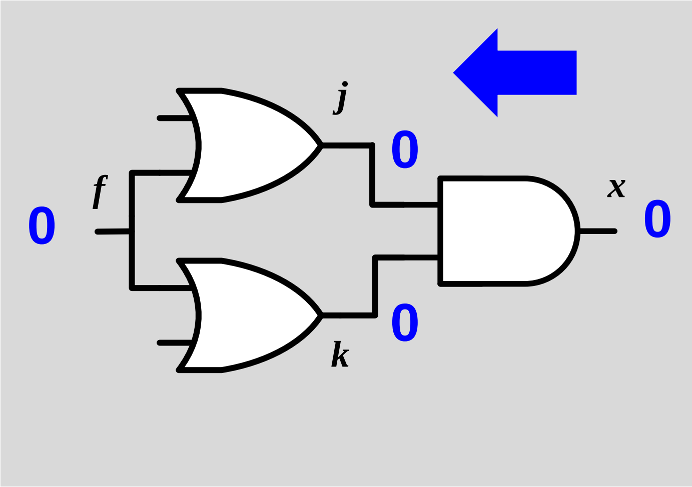


Learning Trade-off Memory for Run Time

But … Too Many to Learn!

- f=1 → j=1 so j=0 → f=0
  - Only local implication. not worth learning
- f=1 → k=1 so k=0 → f=0
  - Only local implication. not worth learning
- f=1 → x=1 so x=0 → f=0
  - global implication. worth learning!





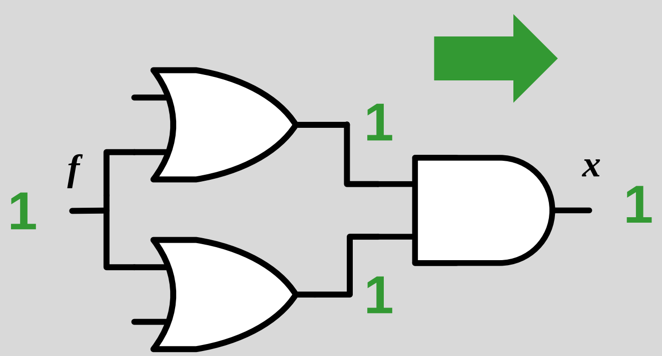
Which Are Worth Learning?

SOCRATES Algorithm [Schulz 1988]

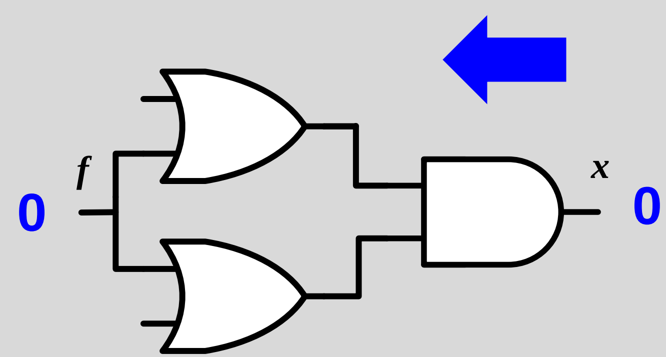
- **In preprocess phase, sets all signals to 0 and 1**
  - **Discovers what other signals are implies**
  - **Two criteria to select useful learning**

```
static_learning()
  for every signal f
    assign_value (f, 0)
    implication()
    analyze_results(f)
    assign_value (f,1)
    implication()
    analyze_results(f)
```

```
analyze_results (f)
  for every signal x whose value ≠ unknown
    if (all gate inputs to x are non-controlling)
    & (there is forward path from f to x)
        save learning result (x=v_x'→ f = v_f')
```



f=1 → x=1

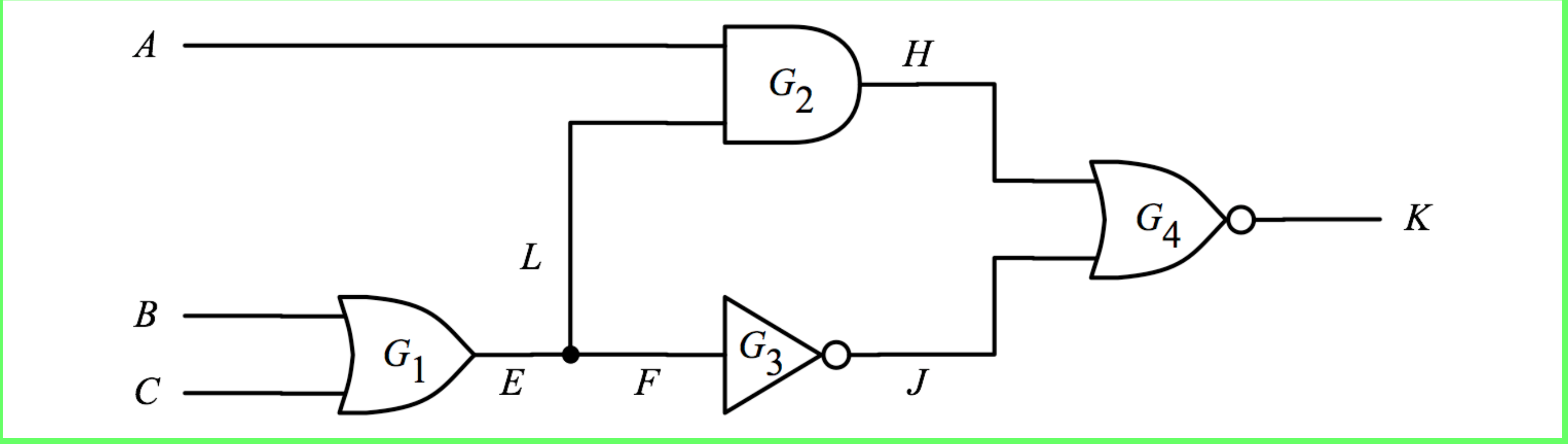


x=0 → f=0

Quiz

Q1: Set *E* = 0. What can you learn about *K* using contrapositive law?
A:

Q2: (Cont'd) If we want to detect *K* SA 0 fault, what is value of *E*?
A:

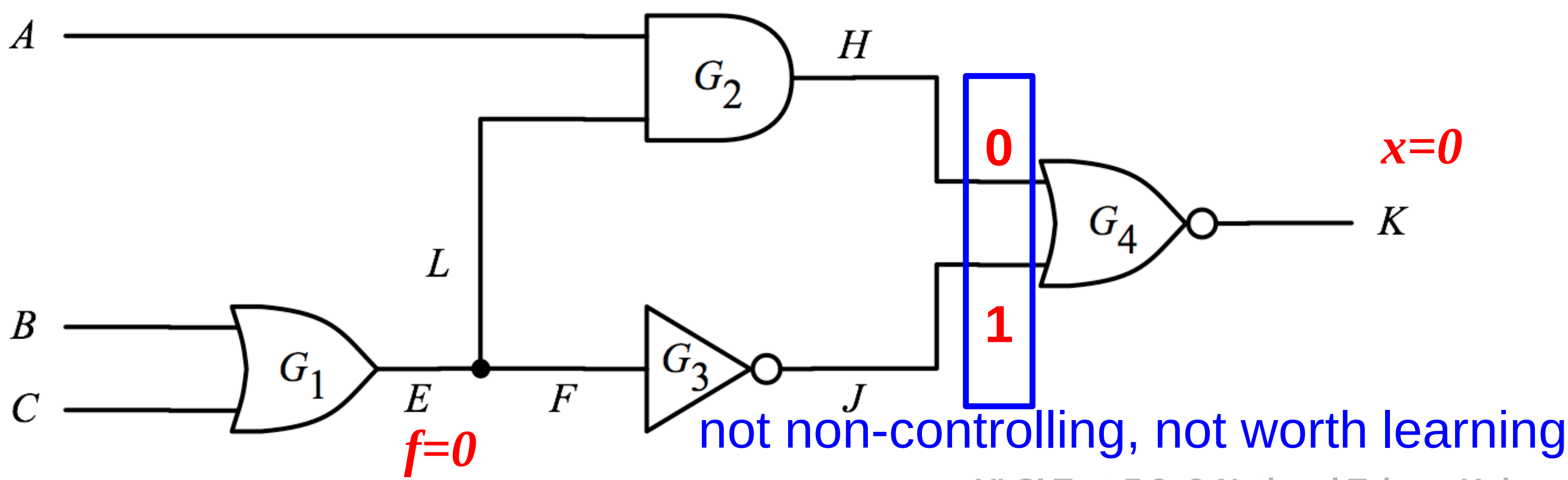


FFT

- **Q1: Why SOCRATES requires both inputs are non-controlling?**
  - **Hint: use following circuit as example**
- **Q2: Why forward path from *f* to *x*?**

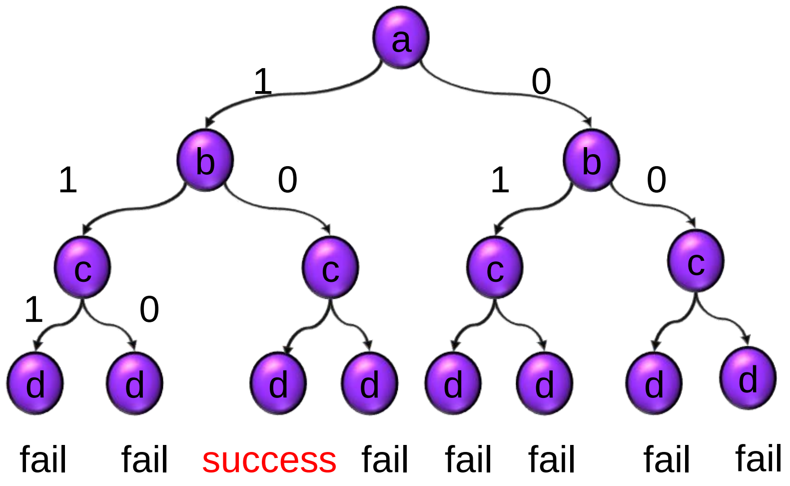
```
static_learning()
  for every signal f
    assign_value (f, 0)
    implication()
    analyze_results(f)
    assign_value (f,1)
    implication()
    analyze_results(f)
```

```
analyze_results (f)
  for every signal x whose value ≠ unknown
    if (all gate inputs to x are non-controlling)
    & (there is forward path from f to x)
        save learning result (x=vx'→ f = vf')
```



Combinational ATPG

- Introduction
- Deterministic Test Pattern Generation
- Acceleration techniques
  - Learning [Schulz 1988]
  - Redundant fault identification [Iyer 1996]
- Concluding Remarks

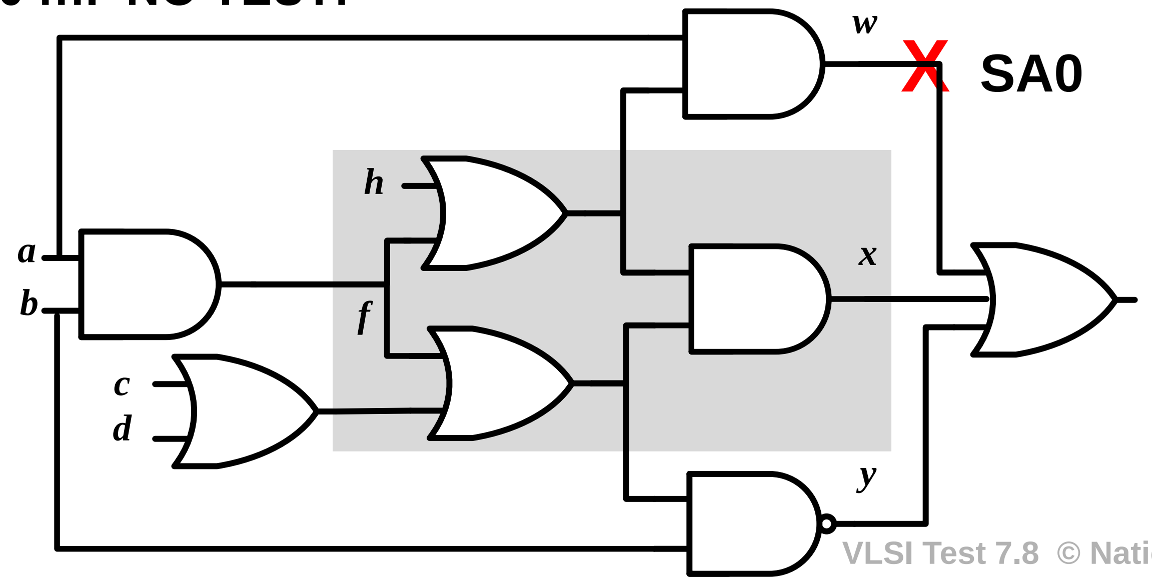


Proving Redundant Fault is Difficult

without learning

- Obj : w=1. assign h=1, a=1
- Obj : x=0. assign c=0, d=0, b=0
  - y=1 conflict!
- backtrack b=1
  - x=1 conflict!
- backtrack c=1, conflict!
- backtrack d=1, conflict!
- backtrack a=0, conflict!
- backtrack h=0 …. NO TEST!

Can We Find Redundant Fault Faster?



Redundant Fault Identification

- *Untestable faults* (aka. *redundant faults*)
  - faults that cannot be excited, or
  - faults that cannot be propagated, or
  - faults that cannot be simultaneously excited and propagated
- Why redundant fault identification?
  1. Speed up ATPG
     - ATPG spend long time on untestable faults
  2. Reduce area
     - Redundant logic can be removed
- To prove redundant fault is NP-complete
  - Quickly identify many redundant fault (not all) is good enough

Redundant Faults are Trouble for ATPG

FIRE [Iyer 1996]

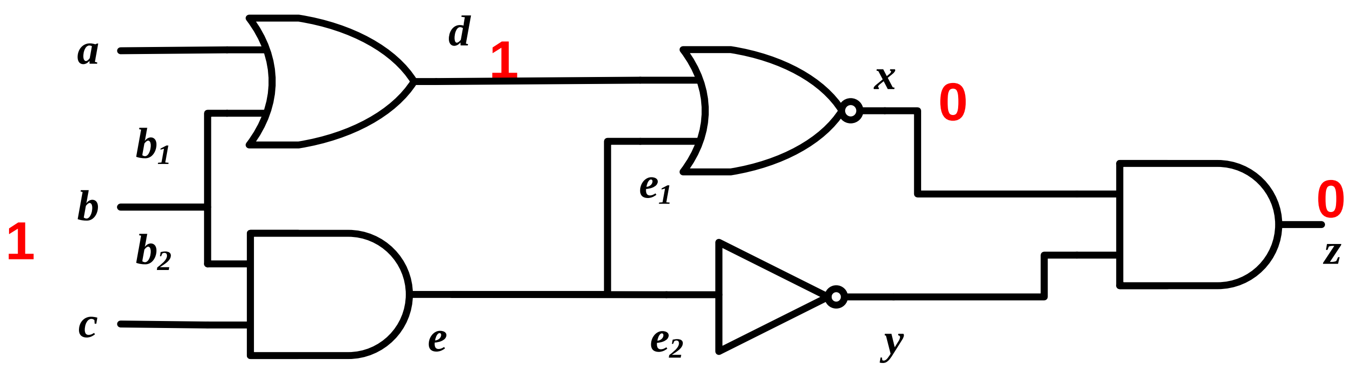
- Fault Independent Redundant Identification (FIRE)
  - based on single-line conflict analysis
- Idea
  - S0 = set of faults untestable when signal s=0
  - S1 = set of faults untestable when signal s=1
  - intersection $S0 \cap S1$ are untestable faults
- To find unexcitable faults
  - use forward implication
- To find unobservable faults
  - use backward tracing

- Advantage: close to linear time complexity (for one signal)
  - FFT: can we solve NPC problem in linear time?

FIRE Example

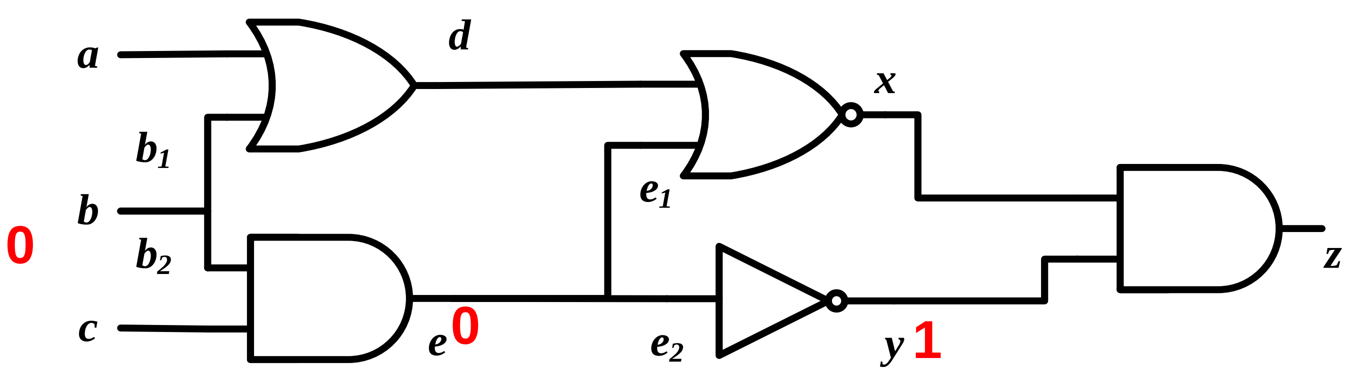
- set b=1 implies → {b=1, b_1=1, b_2=1, d=1, x=0, z=0}
  - Faults unexcitable when b=1: {b/1, b_1/1, b_2/1, d/1, x/0, z/0}
  - Faults unobservable when b=1: {a/0, a/1, e_1/0, e_1/1, y/0, y/1, e_2/0, e_2/1, e/0, e/1, b_2/0, b_2/1, c/0, c/1 }
    - Q: why b/0 not in the list?
  - Faults untestable when b=1: union of above two sets
    - S1={a/0, a/1, b/1, b_1/1, b_2/1, d/1,e1/0, e_1/1, e_2/0, e_2/1, e/1, e/1, x/0, y/0, y/1, z/0, b_2/0, c/0, c/1 }

b/1 = b SA1 fault



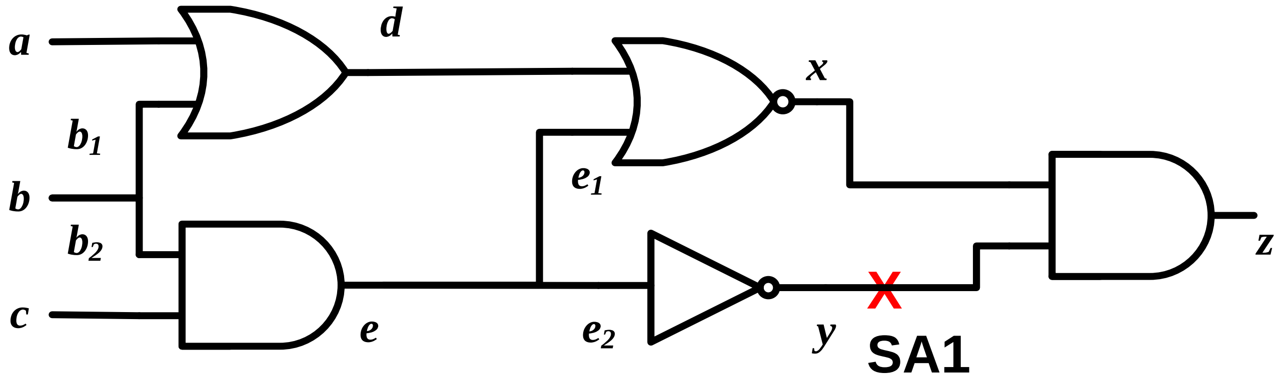
FIRE Example (cont'd)

- set b=0 implies → {b=0, b_1=0, b_2=0, e=0, e_1=0, e_2=0, y=1}
  - Faults unexcitable when b=0: {b/0, b_1/0, b_2/0, e/0, e_1/0, e_2/0, y/1}
  - Faults unobservable when b=0: {c/0, c/1}
  - Faults untestable when b=0: union of above two sets
    - S0= {b/0, b_1/0, b_2/0, c/0, c/1, e/0, e_1/0, e_2/0, y/1}
    - S1={a/0, a/1, b/1, b_1/1, b_2/1, d/1,e1/0, e_1/1, e_2/0, e_2/1, e/1, e/1, x/0, y/0, y/1, z/0, b_2/0, c/0, c/1 }
- Intersection of S1 and S0 are untestable faults
  - S1 ∩ S0 = {b_2/0, c/0, c/1, e/0, e_1/0, e_2/0, y/1}



FFT

- **Q1: Use PODAM to find a test for y/1 fault. Prove y/1 is untestable.**
- **Q2: Can we use linear time algorithm to solve NPC problem?**



Quiz

Q1: Set B=0. Find set of faults undetectable.

A: faults unexcitable = { }

faults unobservable= { }

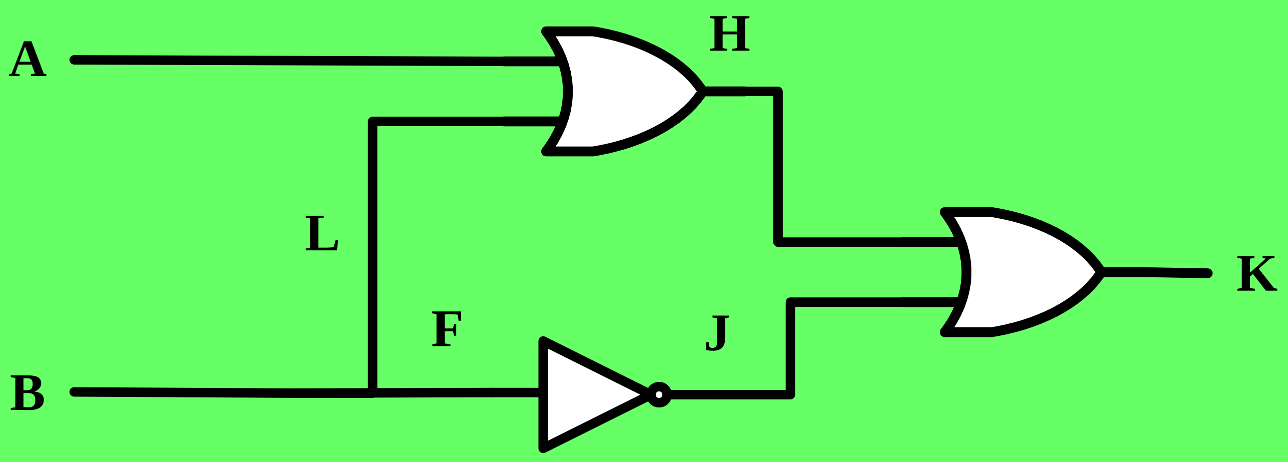
Q2: Set B=1. Find set of faults undetectable.

A: faults unexcitable = { }

faults unobservable= { }

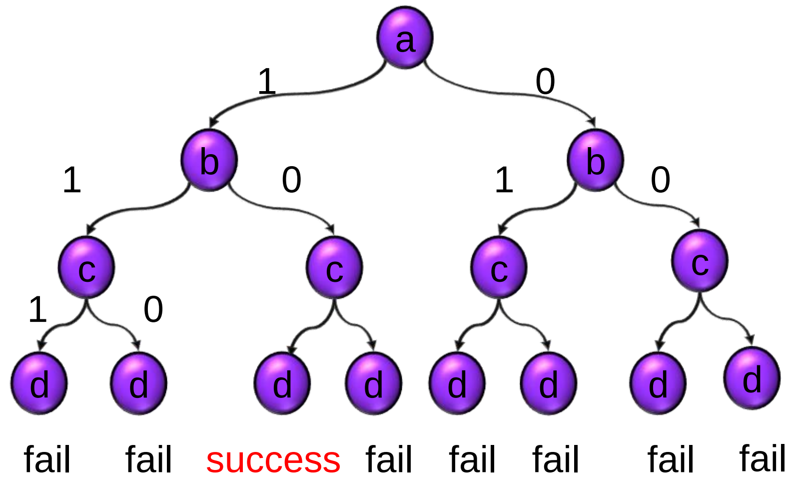
Q3: (cont’d) Which faults are redundant ?

A: Q1$\cap$ Q2={ }



Combinational ATPG

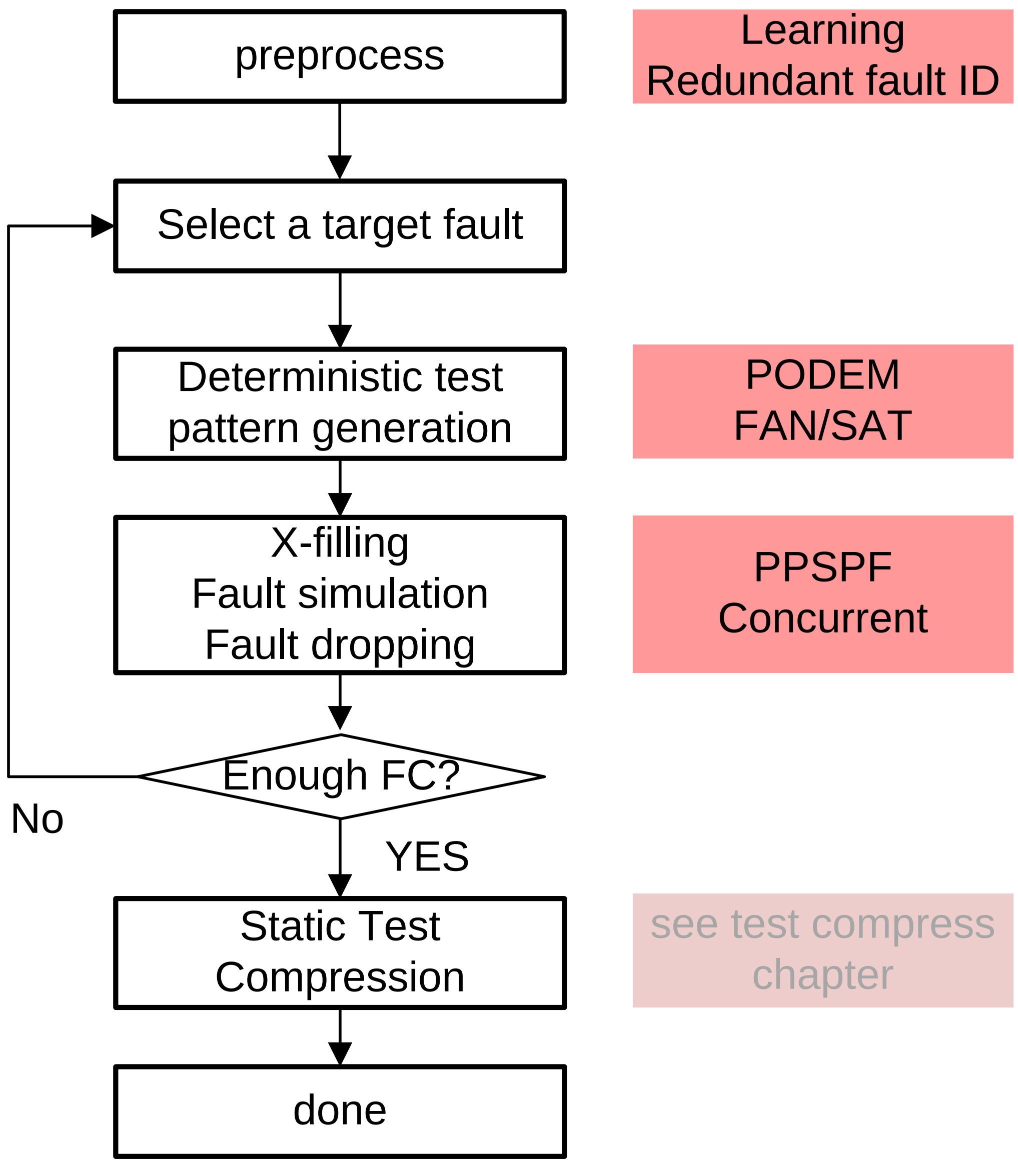
- Introduction
- Deterministic Test Pattern Generation
- Acceleration Techniques
- Concluding Remarks



Summary

- ATPG is NPC. Many acceleration techniques needed
- Static learning (SOCRATES) trade off memory for run time
  - Setting all signals to 0 and 1. imply other signals
  - Apply contrapositive law: if $p \rightarrow q$ then $q' \rightarrow p'$
- Redundant fault identification (FIRE)
  - Setting a signal to 0 and 1
    - unexcitable faults $\cup$ unobservable faults
  - Redundant fault = $S1 \cap S0$
- Many other acceleration techniques
  - Dominator (TOPS) [Kirkland 87]
  - Recursive learning [Kunz 92]
  - Transitive Closure Graph (NNATPG) [Chakradhar 93]
  - …

ATPG Review


preprocess
Learning
Redundant fault ID
Select a target fault
Deterministic test
pattern generation
PODEM
FAN/SAT
X-filling
Fault simulation
Fault dropping
PPSPF
Concurrent
Enough FC?
No
YES
Static Test
Compression
see test compress
chapter
done

How to Read ATPG Report?

	Uncollapsed	Collapsed
Total Faults	1234	800
Detected faults	1000	700
Redundant faults	230	98
Aborted faults	4	2
Fault coverage	1000/1234 %	700/800 %
ATPG effectiveness	1230/1234 %	798/800 %
Test Length	328 patterns	
Run Time	10:57	

proven redundant by ATPG

undetected. not sure redundant or not.

$$\frac{detected\ \ faults}{total\ \ faults} \times 100\%$$

$$\frac{detected + redundant\ \ faults}{total\ \ faults} \times 100\%$$

less are better

References

- [Fujiwara 83] H. Fujiwara and T. Shimono, “On the Acceleration of Test Generation Algorithms, “ Proc. Int’l Fault-Tolerance Computing Symp., pp.98-105, 1983.
- [Goel 81] P. Goel, “An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits,” IEEE Trans. On Computers, Vol. C-30, No. 3, pp.215-222. Mar. 1981.
- [Roth 66] J. P. Roth, “Diagnosis of Automata Failures: A Calculus and a Method,” IBM Journal of Research and Development, vol. 10, no. 4, pp278-291, 1966.
- [Larrabee 92] T. Larrabee, “Test pattern generation using Boolean satisfiability,” IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, Volume 11, Issue 1, Jan 1992, pp. 4-15.
- [Iyer 96] Iyer, M.A. Abramovici, M. “FIRE: a fault-independent combinational redundancy identification algorithm,” Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, Jun 1996, Volume: 4, Issue: 2 page(s): 295-301
- [Schulz 88] M.H Schulz, et al “SOCRATES: A highly efficient automatic test pattern generation system IEEE TCAD, vol.7 no.1 pp.126, 1988.
- [Marques 94] J. Marques, S. Karen, A. Sakallah, “Dynamic search-space Pruning Techniques in Path Sensitization’ DAC 1994.

Commercial Tools

- Mentor Graphics
  - Fastscan
- Synoposys
  - Tetramax
- Syntest
  - Turboscan
- Cadence
  - Encounter Test