CIS 4930 Digital System Testing
Fault Simulation

Dr Hao Zheng
Comp. Sci & Eng.
U of South Florida

Overview

- Fault simulation applications
- Fault simulation techniques
  - Serial
  - Parallel
  - Deductive
  - Concurrent
- **tentative**
  - Fault simulation for combinational circuits
  - Fault sampling
  - Statistical fault analysis

5.1 Applications

Fault Simulation

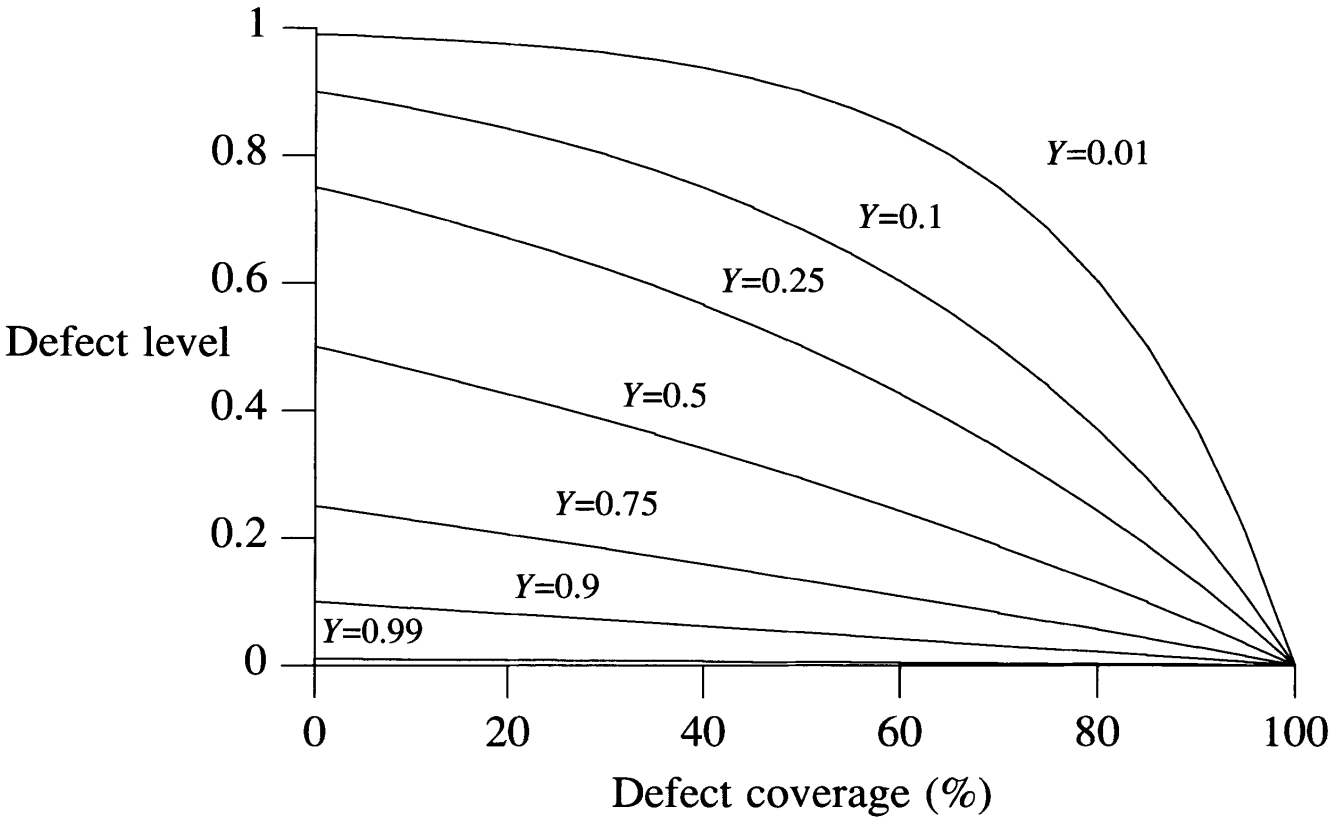
- Simulation of a circuit in the presence of faults
- Used to
  - Evaluate a test *T wrt* fault coverage.
  - Generate tests *T* to achieve certain fault coverage.
  - Construct fault dictionary
  - Analyze circuit operation in the presence of faults

1 – Evaluate a Test *T*

- Usual metric: fault coverage
- Fault coverage relevant to the fault model
  - 100% FC does not mean 100% defects are covered if the fault model is limited.
  - Other defects may still exists if not considered in a fault model.
  - Lower bound on **defect coverage**
- Defect coverage ***d*** = probability that *T* detect any physical fault.
  - Has a big impact on product quality.

Yield and Defect Level

- **Defect level** (DL) = prob. of shipping a defective product
- **Yield** (Y) = prob. that manufactured circuit is defect free



$$DL = 1 - Y^{1-d}$$

2 – Test Evaluation

→ Enhance T until adequate fault coverage is satisfactory

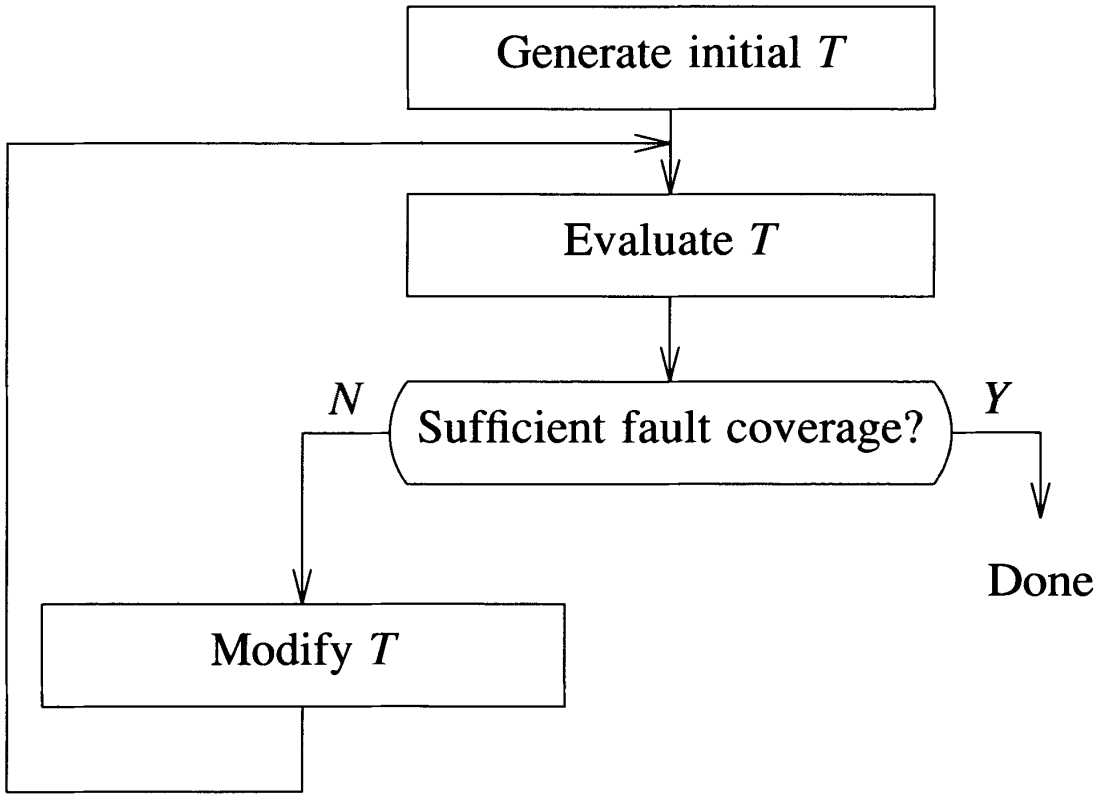
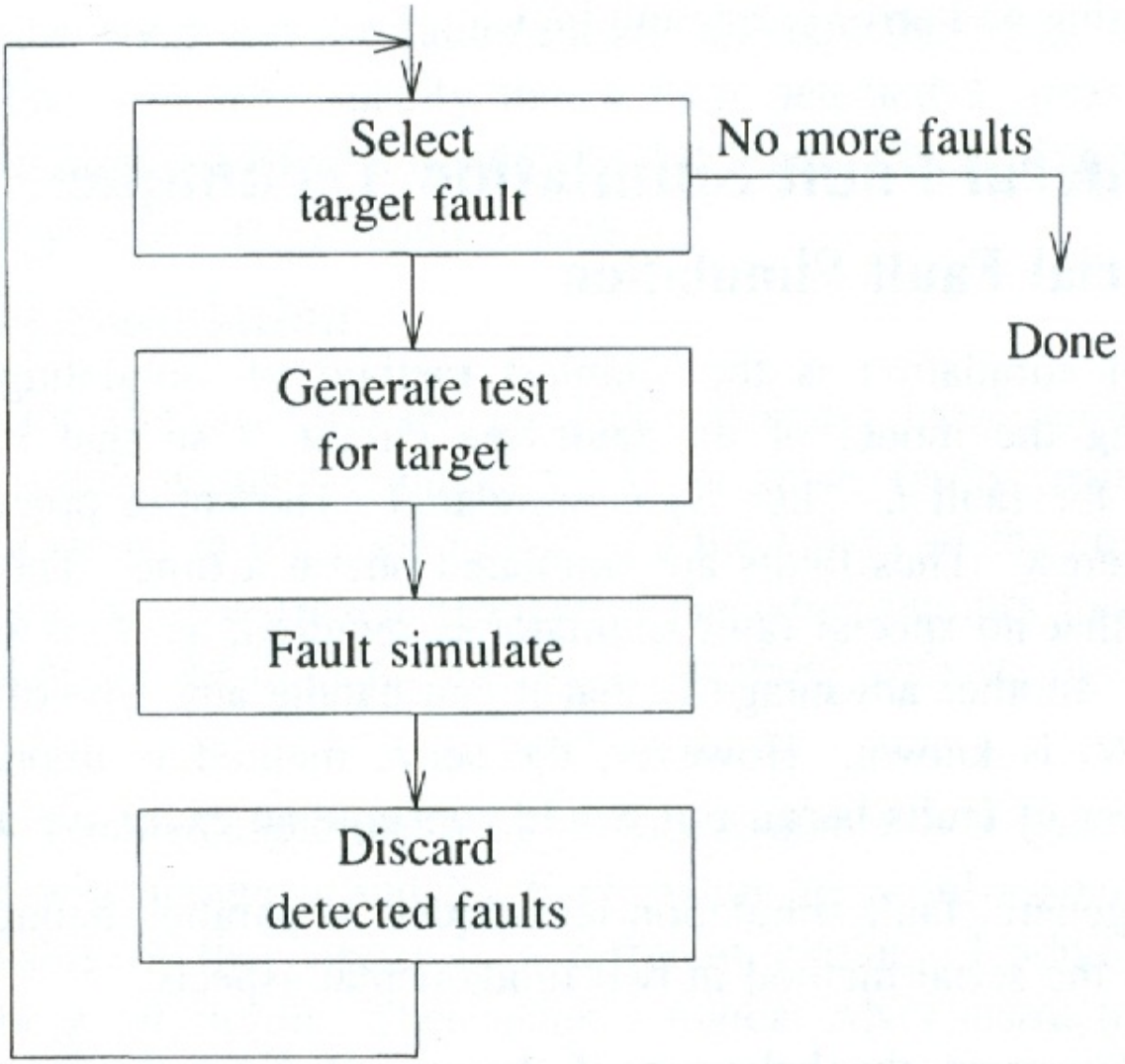


Figure 5.2 General use of fault simulation in test generation

Test Generation

Target fault oriented approach



3 – Construct Fault Dictionaries

→ Fault Dictionary – stores output response (R_f) or *signature* $S(R_f)$ to *T* of every faulty circuit N_f

	f1	f2	..	fn
T1	0	1	..	1
T2	1	0	..	1
:	:	:		:
Tm	1	1	..	0

4 – Circuit Analysis

- Analyze circuit operations in presence of faults
- Some effects introduced by faults may not present in fault-free circuit:
  - Races and/or hazards
  - Oscillation and/or deadlock
  - Inhibit proper initialization of seq. circuit
  - Transform combinational to sequential
  - Transform synchronous to asynchronous

5.2 Fault Simulation Techniques

General Fault Simulation Techniques

- Serial Fault Simulation
- Parallel Fault Simulation
- Deductive Fault Simulation
- Concurrent Fault Simulation

Serial Fault Simulation

- Simulate faults one at a time
- Given a fault f, do the following:
  - Transform N to N_f
  - Simulate N_f
- Repeat for other faults under consideration.
- Advantage
  - No need for a special fault simulator
- Disadvantage
  - Impractical for large number of faults

Other Three Techniques

- Common characteristics:
  - Do not change the circuit model
  - Can simultaneously simulate a *set of faults(!)*
  - Simultaneously simulate *good* and *bad* circuits
- **One-Pass** – If all faults are simulated simultaneously
- **Multi-Pass** – For large set of faults, need multiple simulation runs

Tasks in Fault Simulation

- **Fault specification**: define set of modeled faults and perform fault collapsing
- **Fault insertion:** select a fault subset and create data structures to indicate fault presence.
- **Fault effect generation:** Say line *i* has *f s-a-1* then whenever value 0 propagates on line *i,* then simulator changes it to 1
- **Fault effect propagation:** Propagate v/v_f to primary output for fault detection
- **Fault discarding:** Inverse of fault insertion
  - Discard a fault if it is detected for *k* times.

Parallel Fault Simulation

- Simultaneously simulate the good circuit and *W* copies of faulty circuits
- Set F of faults needs $\lceil F/W \rceil$ number of passes
- Values of the same signal in different circuits are packed into one memory location (a word or multi-words).

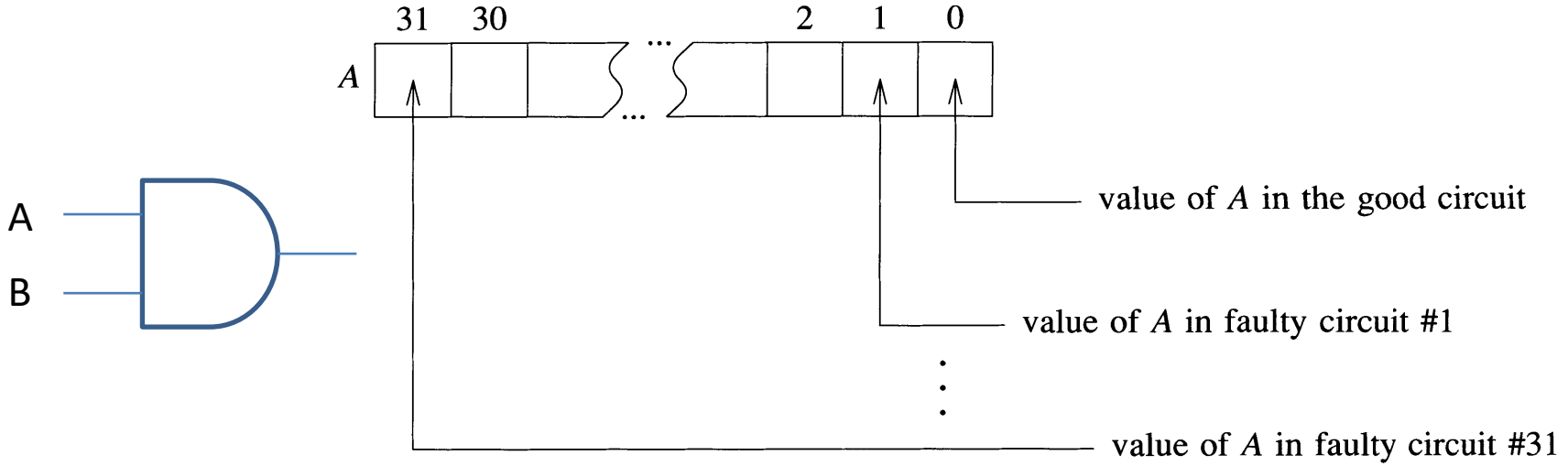


Figure 5.4 Value representation in parallel simulation

Function Evaluation

- Words for A and B are bitwise ANDed (for eg.) for logic AND.
- Similar for other Boolean operations.
- Sequential circuit: For eg., JK FF

$$Q^+ = J\overline{Q} + \overline{K}Q$$

$$Q = clk\uparrow \ ?\ Q^+ : Q$$

The above expression is a Boolean expression consisting of AND, OR, and NOT

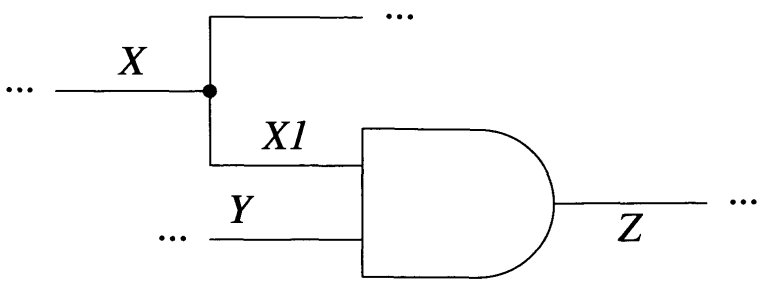
Bit Value Computation

- Let v_i be the value on line i in the faulty circuit N_f where *f* is the fault *j s-a-c*
- Then,

$$v_i' = v_i \overline{\delta_{ij}} + c\delta_{ij}$$

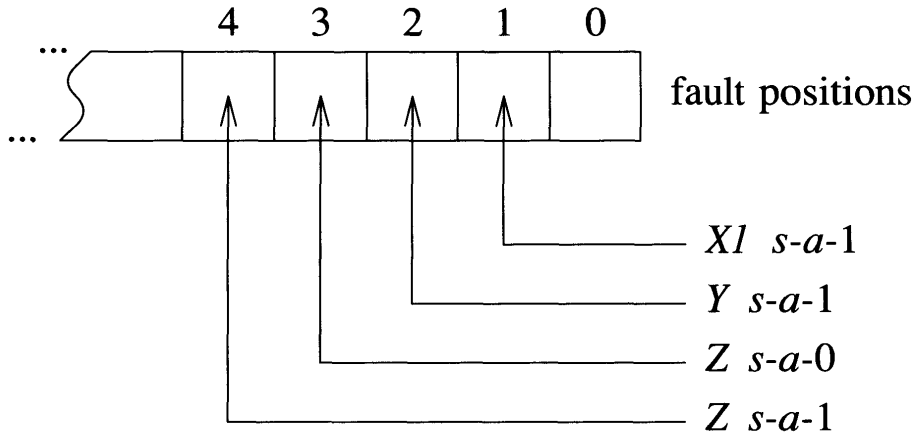
$$\text{where } \delta_{ij} = \begin{cases} 0 & i \neq j \\ 1 & i = j \end{cases}$$

Fault insertion for one fault

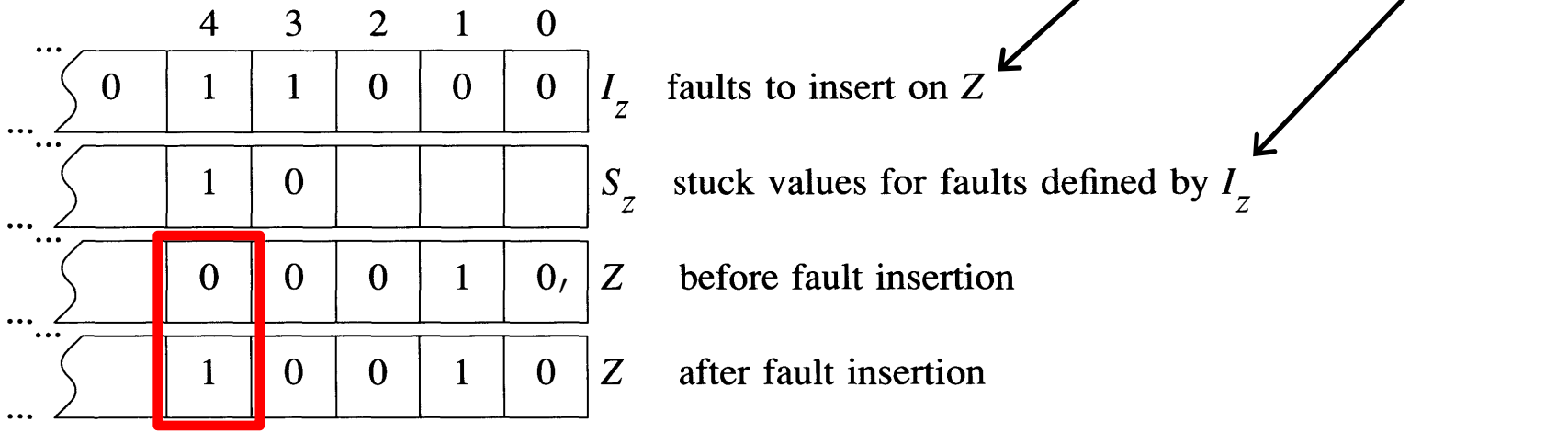


$$Z = X_1 \cdot Y$$

$$Z' = Z\overline{I_Z} + S_Z I_Z$$



To discard a fault simply make the it's mask bit = 0

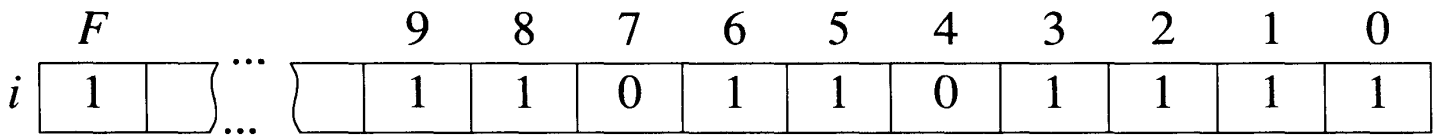


Parallel FS - Limitations

- Parallel simulation is limited for functional level modeling
  - For example if we have to examine for a word value, we need to extract the bits and then re-pack
- Impractical for multi-valued logic
- Event on one bit position results in enter word evaluation => wasted computation
- Cannot take advantage of fault dropping
  - Even if all but one faults are dropped, we still evaluate *W* copies!

Deductive Fault Simulation

- Simulates good circuit and deduces the behavior of *all* faulty circuits (limited by memory)
- Maintains ***Fault List,*** L_i for each signal line *i.*
- L_i = List of all faults *f* that cause the values on *i* in *N* and N_f to be different at the current simulation time
- Difference with Parallel Simulation:



$L_i = \{4,7\}$

Figure 5.8 Fault-effects representation in parallel and deductive fault simulation

How Deductive Simulation Works

- Given
  - Fault-free input values, and
  - Fault lists on inputs of an element
- Compute:
  - Fault-free output
  - Output fault list (i.e., fault list propagation)

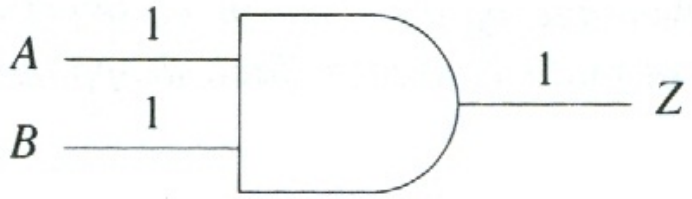
Two Valued Deductive Simulation

- Any fault that causes A or B = 0 will lead to Z = 0
- Therefore:

$$L_Z = L_A \cup L_B \cup \{ Z \text{ s-a-0} \}$$

$$L_A = \{A \ s-a-0\}$$

$$L_B = \{B \ s-a-0\}$$

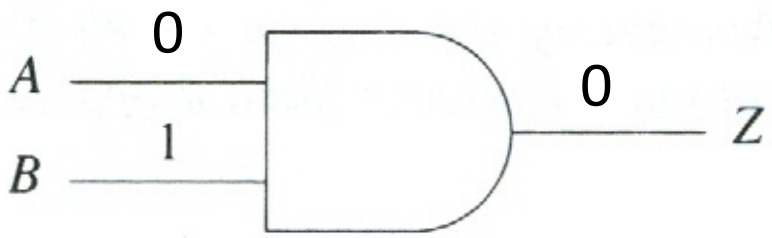


Use *Ax* to denote *A s-a-x*

Two Valued Deductive Simulation

- Any fault that causes A = 1 without changing B, will cause an error on Z
- Note -- A fault that propagates on both A and B will not affect Z
- Therefore:

$$L_Z = (L_A \cap L_B) \cup \{ Z_1 \}$$
$$= (L_A - L_B) \cup \{ Z_1 \}$$



General Formulae

→ Let *I* = set of inputs of gate Z

C = set of inputs with control value *c*

Then Fault List L_Z on *Z* is given by

if $C = \Phi$ **then**

$$L_Z = \{\bigcup_{j \in I} L_j\} \cup \{Z \ s-a-(c \oplus i)\}$$

else

$$L_Z = \{\bigcap_{j \in C} L_j\} - \{\bigcup_{j \in I-C} L_j\} \cup \{Z \ s\text{-}a\text{-}(\overline{c} \oplus i)\}$$

Example

After Fault Collapsing, the fault set is

$$F = \{ a_0, a_1, b_1, c_0, c_1, d_1, e_0, g_0, h_0, h_1 \}$$

Assume T = 00110 to *abcde*

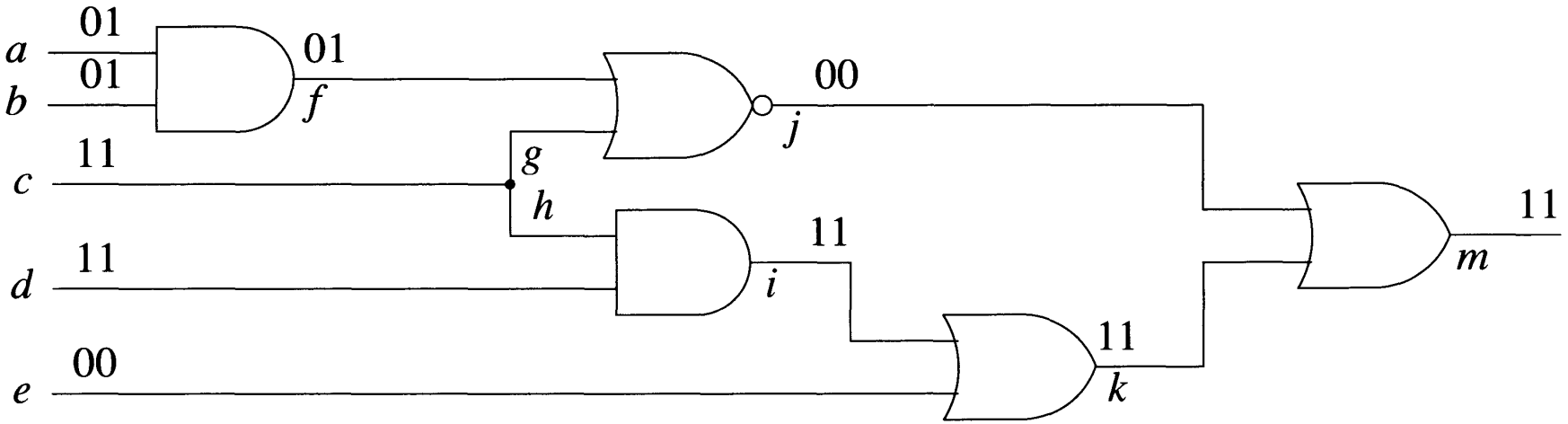


Figure 5.10

$F = \{ a_0, a_1, b_1, c_0, c_1, d_1, e_0, g_0, h_0, h_1 \}$

$L_a = \{a_1\}$, $L_b = \{b_1\}$, $L_c = \{c_0\}$, $L_d = \varnothing$, $L_e = \varnothing$

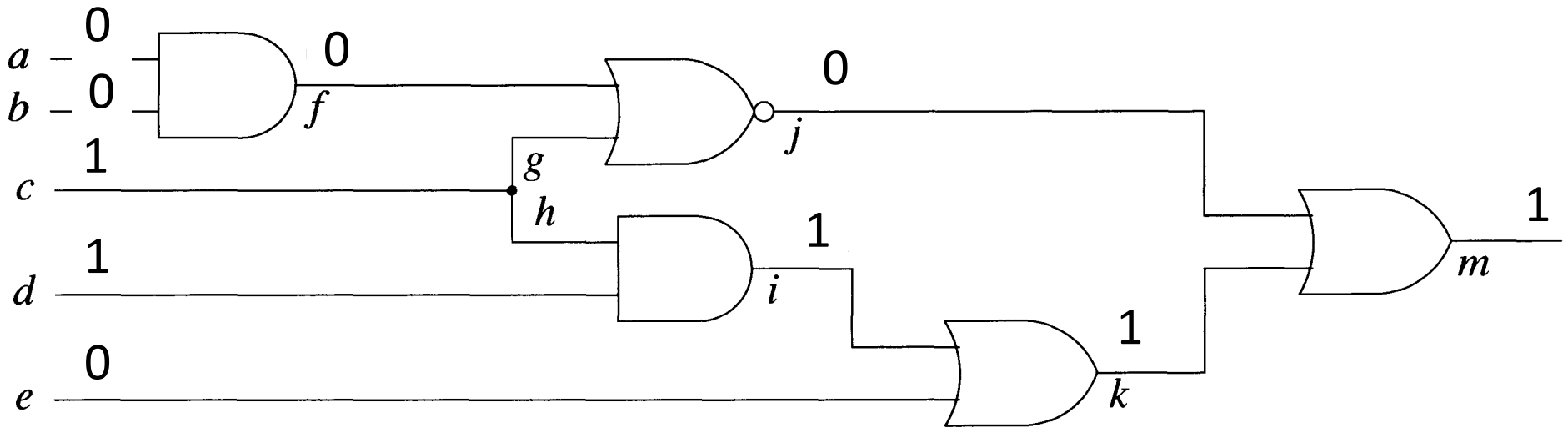


Figure 5.10

$F = \{ a_0, a_1, b_1, c_0, c_1, d_1, e_0, g_0, h_0, h_1 \}$

$L_a = \{a_1\}$, $L_b = \{b_1\}$, $L_c = \{c_0\}$, $L_d = \varnothing$, $L_e = \varnothing$

$L_f = L_a \cap L_b = \varnothing$,

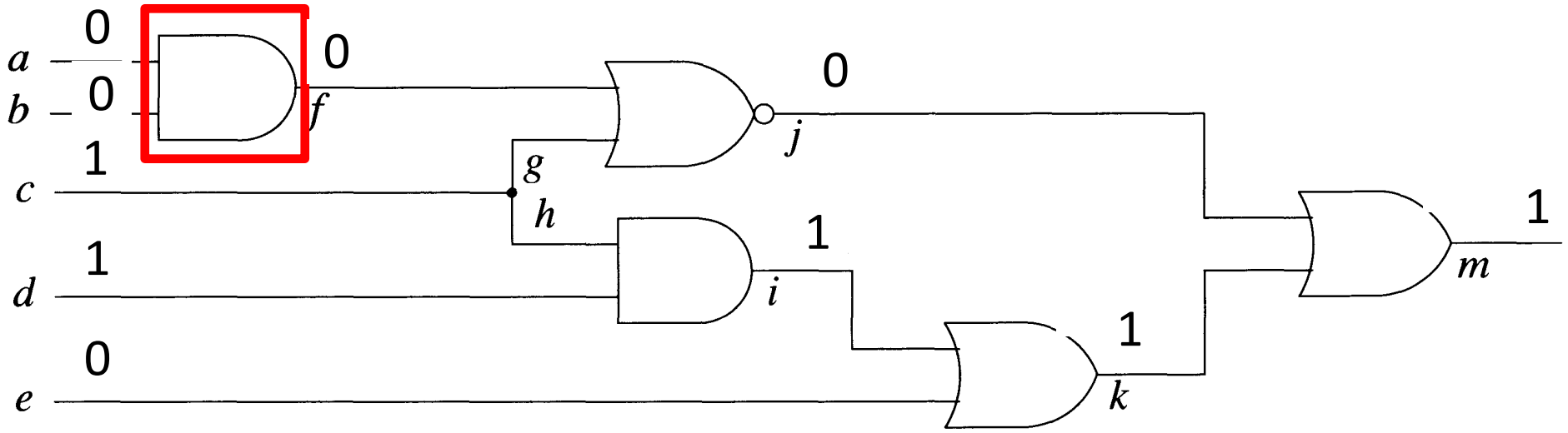


Figure 5.10

$F = \{ a_0, a_1, b_1, c_0, c_1, d_1, e_0, g_0, h_0, h_1 \}$

$L_a = \{a_1\}$, $L_b = \{b_1\}$, $L_c = \{c_0\}$, $L_d = \varnothing$, $L_e = \varnothing$

$L_f = L_a \cap L_b = \varnothing$, $L_g = L_c \cup \{g_0\} = \{c_0, g_0\}$

$L_h = L_c \cup \{h_0\} = \{c_0, h_0\}$

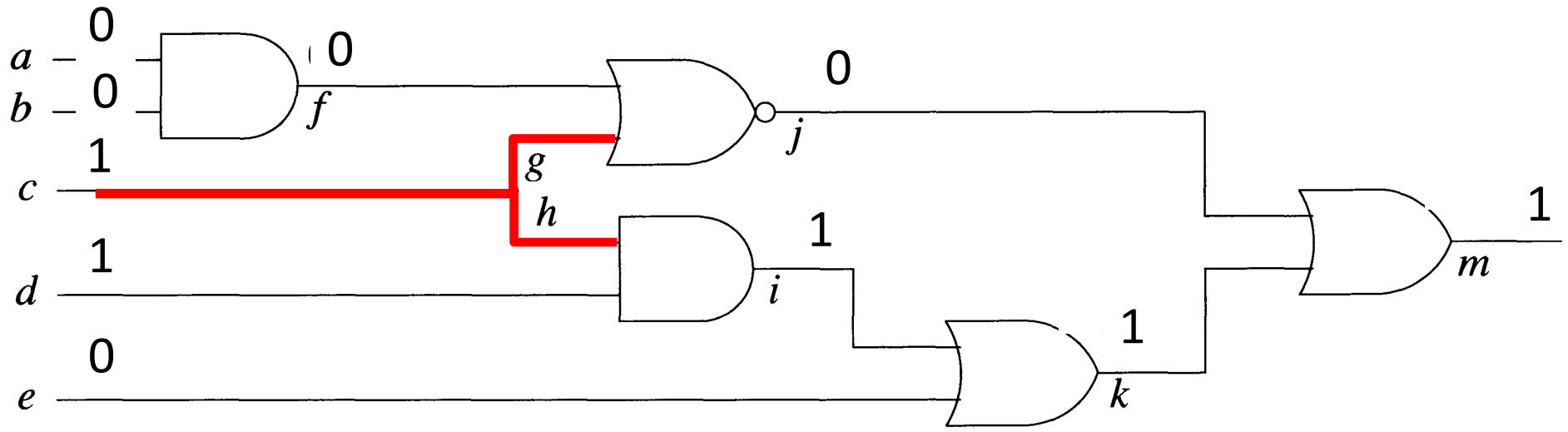


Figure 5.10

$F = \{ a_0, a_1, b_1, c_0, c_1, d_1, e_0, g_0, h_0, h_1\}$

$L_a = \{a_1\}$, $L_b = \{b_1\}$, $L_c = \{c_0\}$, $L_d = \varnothing$, $L_e = \varnothing$

$L_f = L_a \cap L_b = \varnothing$, $L_g = L_c \cup \{g_0\} = \{c_0, g_0\}$

$L_h = L_c \cup \{h_0\} = \{c_0, h_0\}$, $L_j = L_g \cup L_f = \{c_0, g_0\}$

$L_i = L_d \cup L_h = \{c_0, h_0\}$ $L_k = L_i - L_e = \{c_0, h_0\}$

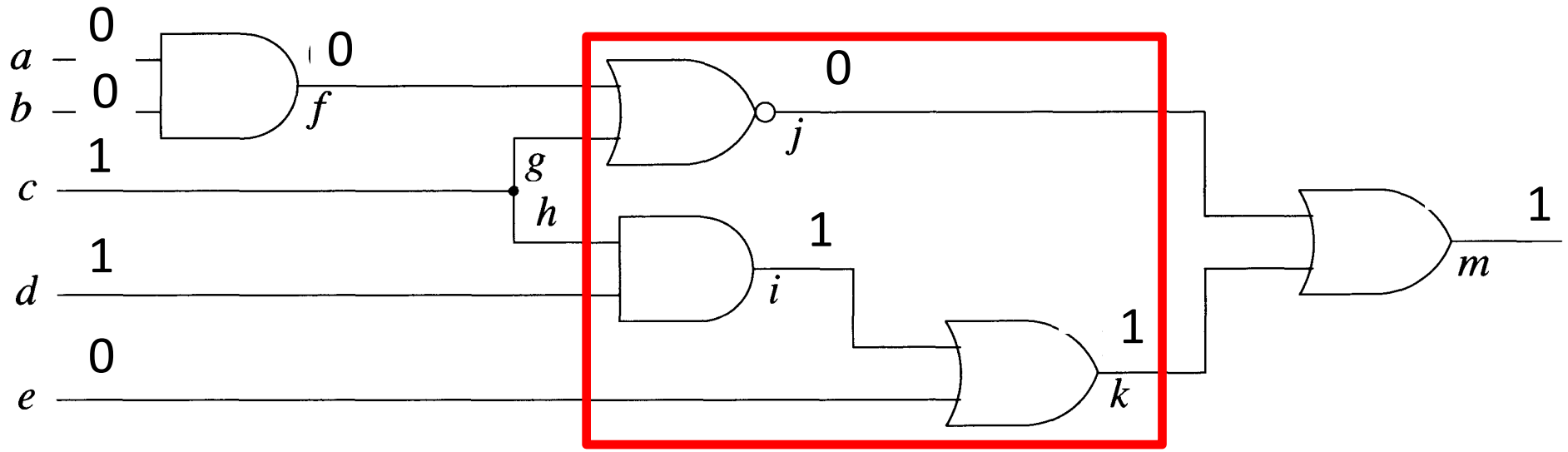


Figure 5.10

$F = \{ a_0, a_1, b_1, c_0, c_1, d_1, e_0, g_0, h_0, h_1\}$

$L_a = \{a_1\}$ $L_b = \{b_1\}$ $L_c = \{c_0\}$ $L_d = \varnothing$ $L_e = \varnothing$

$L_f = L_a \cap L_b = \varnothing$ $L_g = L_c \cup \{g_0\} = \{c_0, g_0\}$

$L_h = L_c \cup \{h_0\} = \{c_0, h_0\}$. $L_j = L_g - L_f = \{c_0, g_0\}$

$L_i = L_d \cup L_h = \{c_0, h_0\}$ $L_k = L_i - L_e = \{c_0, h_0\}$

$L_m = L_k - L_j = \{h_0\}$

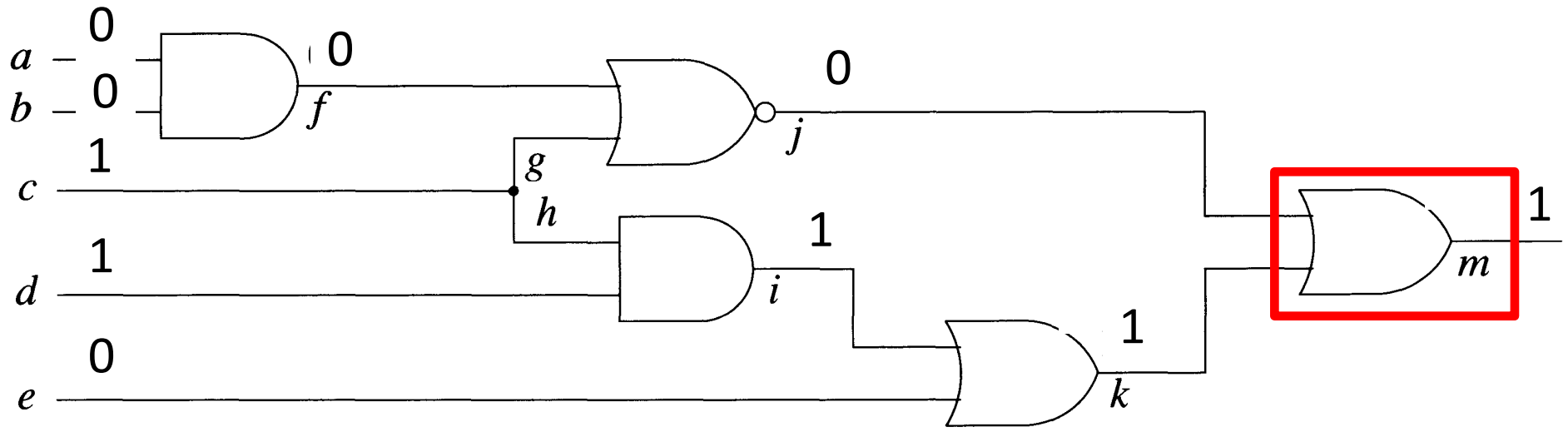


Figure 5.10

Now assume that next test vector is 11110. Redo the example.

L_a = L_b = L_c = L_d= L_e =

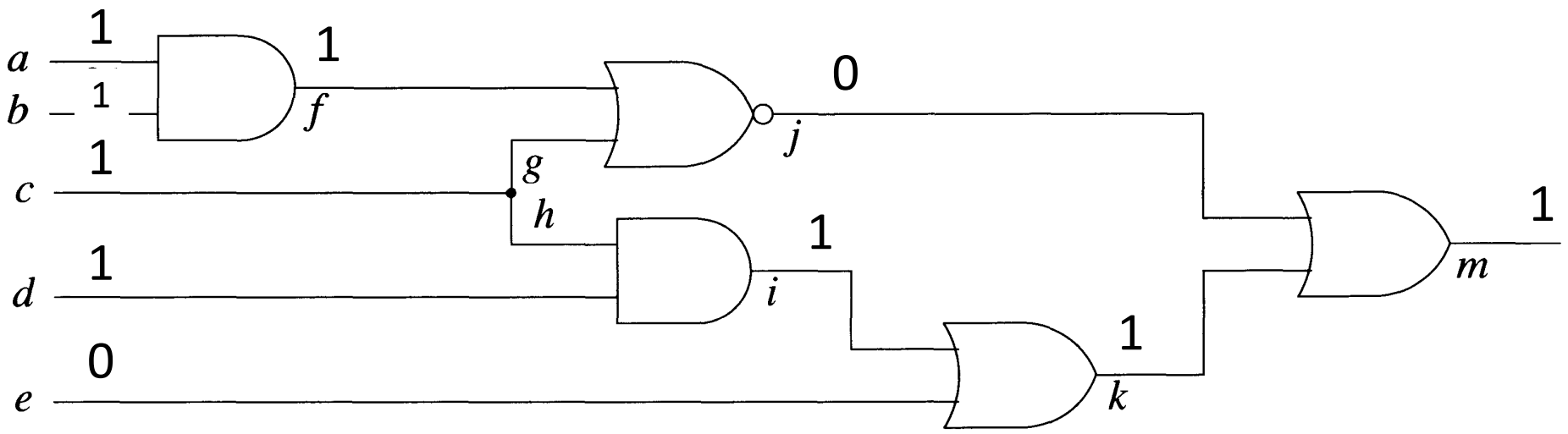
L_f = L_g =

L_h =

L_j = L_i =

L_k = L_m =

F = { a_0, a_1, b_1, c_0, c_1, d_1, e_0, g_0, h_0, h_1}



Solution:

$L_a = \{a_0\}$ $L_b = \varnothing$ $L_c = \{c_0\}$ $L_d = \varnothing$ $L_e = \varnothing$

$L_f = L_a \cup L_b = \{a_0\}$ $L_g = L_c \cup \{g_0\} = \{c_0, g_0\}$

$L_h = L_c \cup \{h_0\} = \{c_0, h_0\}$

$L_j = L_f \cap L_g = \varnothing$ $L_i = L_d \cup L_h = \{c_0, h_0\}$

$L_k = L_i - L_e = \{c_0, h_0\}$ $L_m = L_k - L_j = \{c_0, h_0\}$

Fault c_0 is detected!

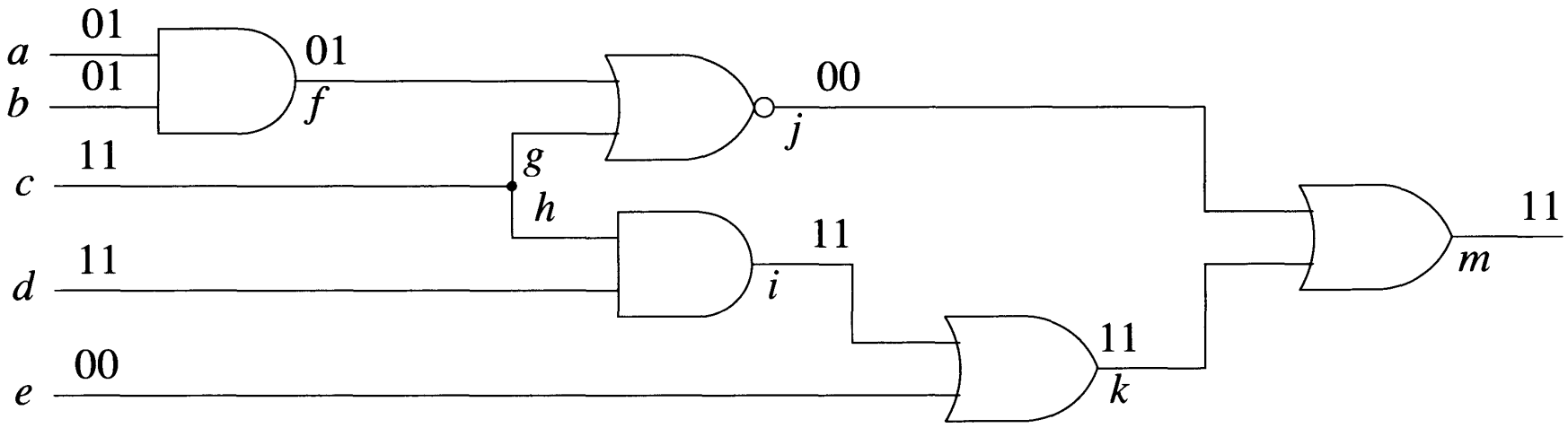


Figure 5.10

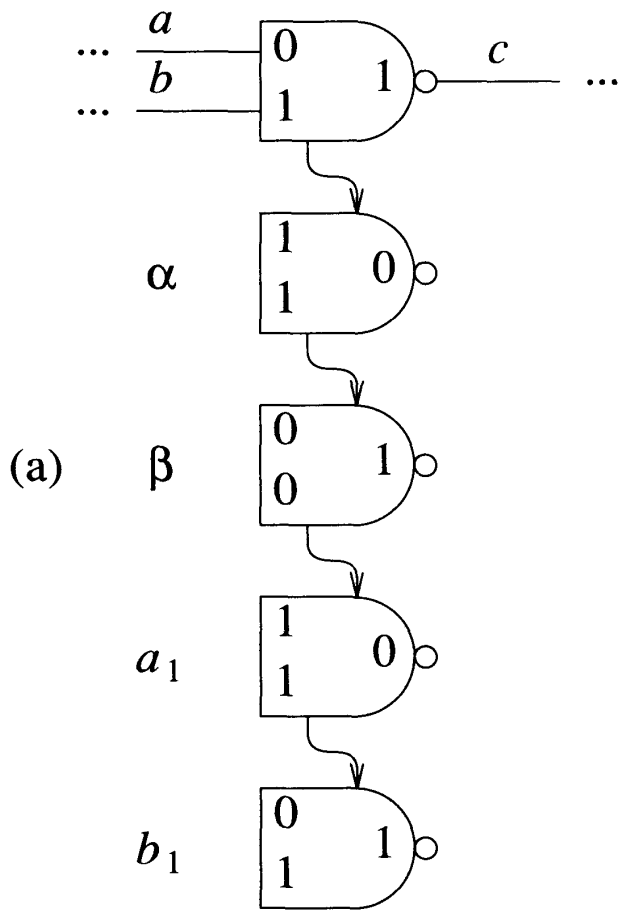
DS - Limitations

- Compatible only in part with functional level modeling
  - Applicable only to models with Boolean eqns.
- Limited to two or three logic values
- Cannot handle timing models
- Fault propagation mechanism cannot take full advantage of the concept of activity-directed simulation

Concurrent Fault Simulation

- Observation – Most of the time, most values in most fault circuits agree with those in the good circuit.
- Concurrent Method
  - simulates the good circuit *N*
  - For every faulty circuit N_f *simulate only those elements that differ with corresponding ones in N*
  - The differences of an element x in *N* is stored as a **concurrent fault list** (CL_x)

Concurrent List Example

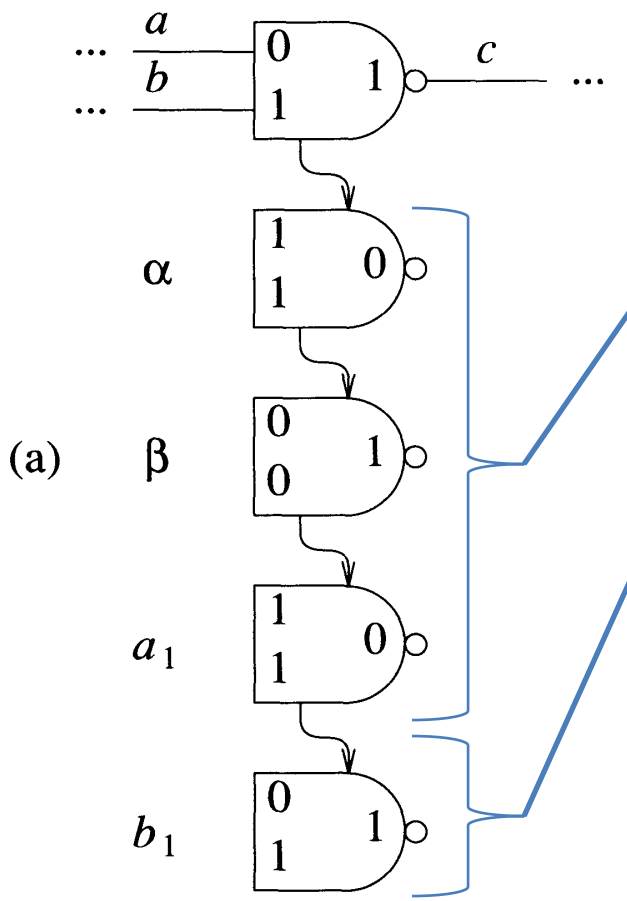


f	a	b	c
α	1	1	0
β	0	0	1
a_1	1	1	0
b_1	0	1	1

(b)

Figure 5.15 Concurrent fault list for gate c (a) Pictorial representation (b) Tabular representation

Two Cases of Differences



Let x_f be replica of x in N_f

V_{xf} and V_x be <inputs, output>

Case 1: $V_{xf} \neq V_x$

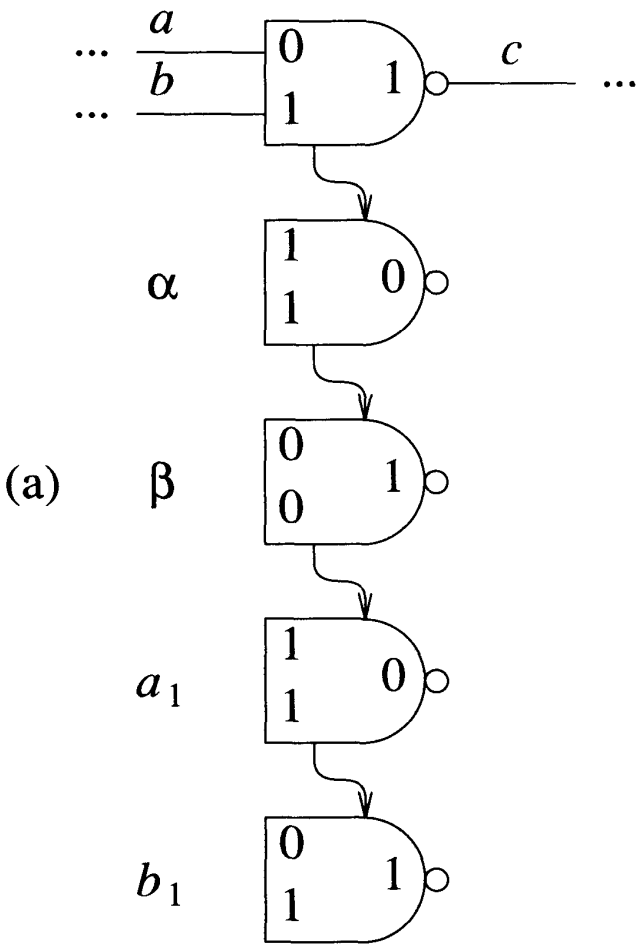
This happens when fault effect of f propagates to x

Case 2: $V_{xf} = V_x$

This happens when f is a local fault (i.e., input/output fault)

Note: Even if the V_x and V_{xf} are equal, the elements are different because of the local fault.

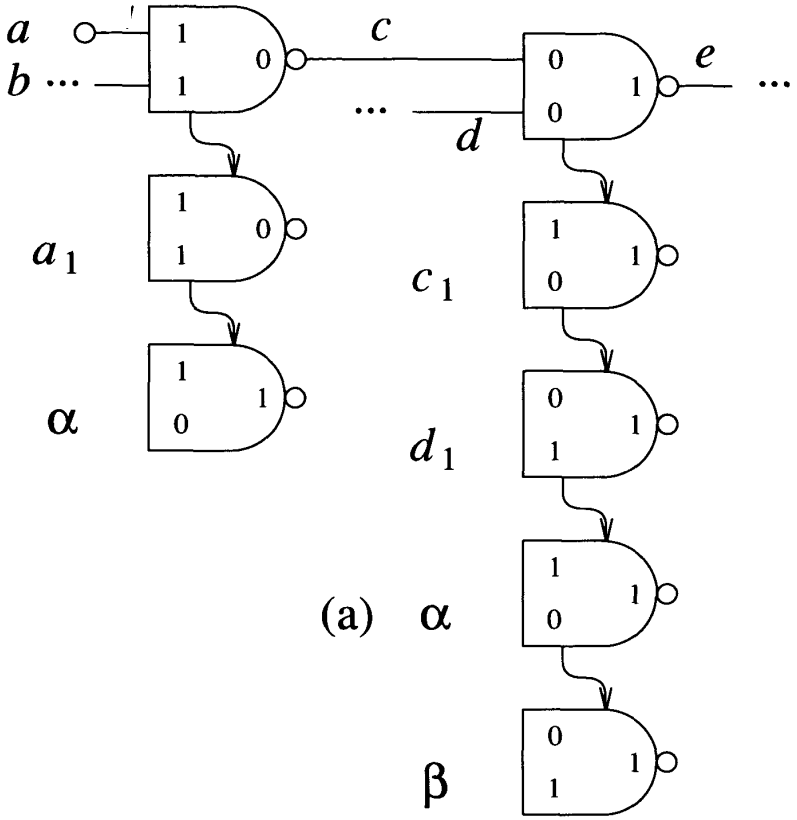
Visible Faults



A fault is visible on line *i* when the values of *i* in *N* and N_f are different.

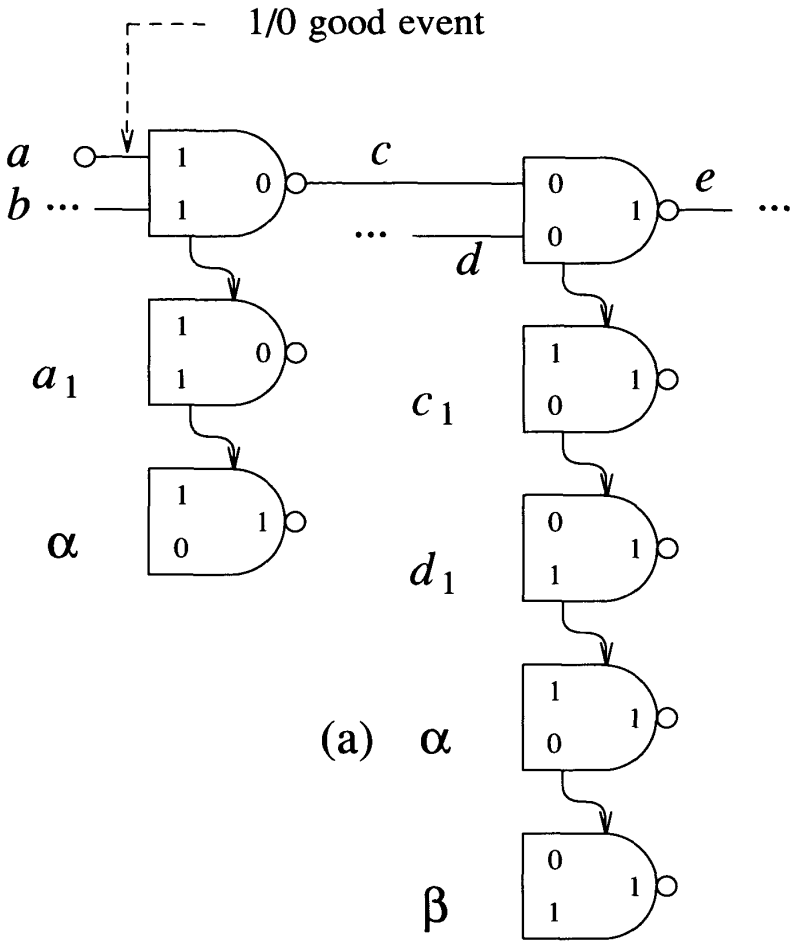
A deductive fault list includes all visible faults, which is subset of the concurrent fault list.

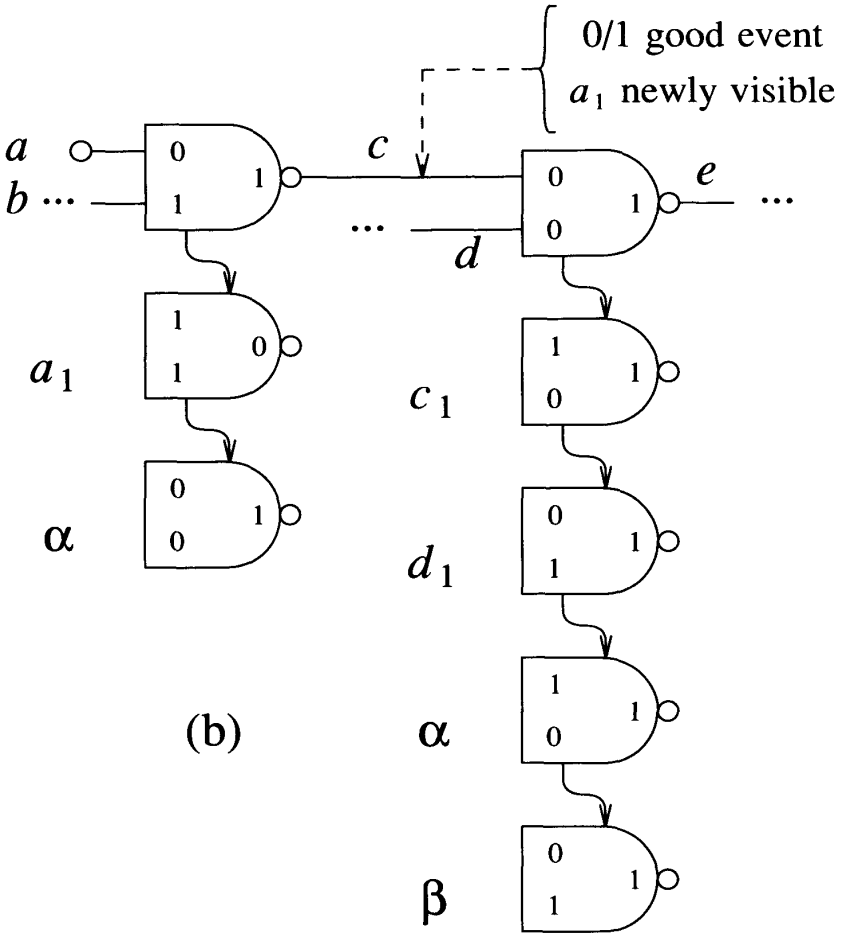
CFS - Example



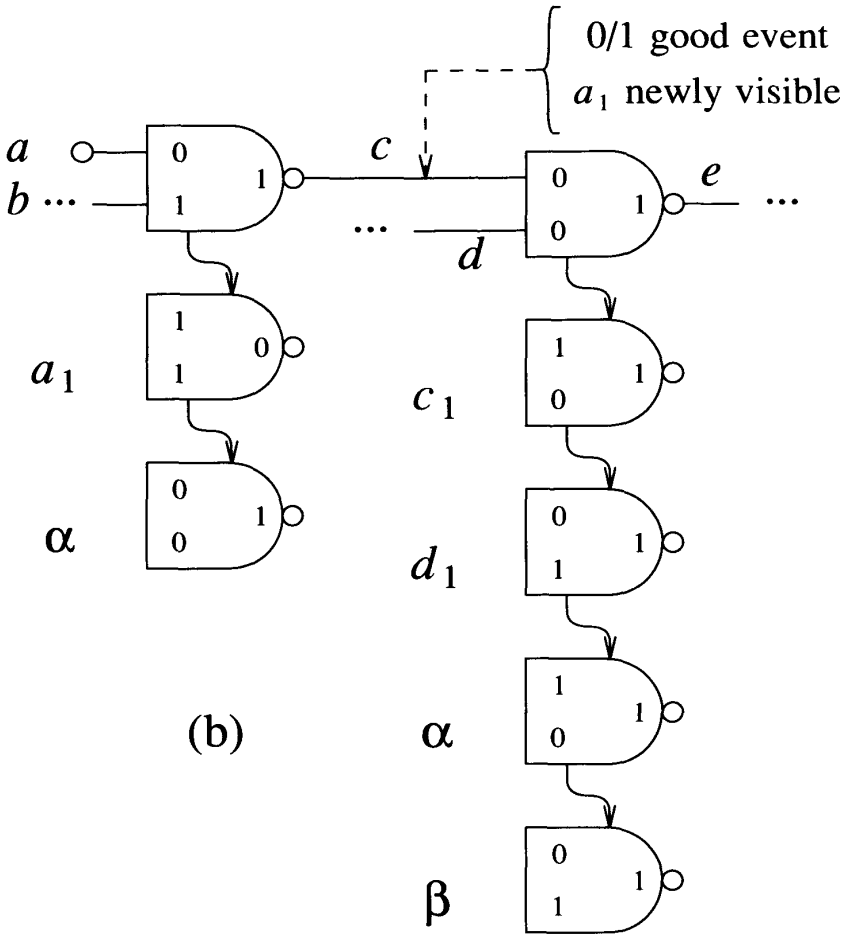
What are those faults in the initial state?

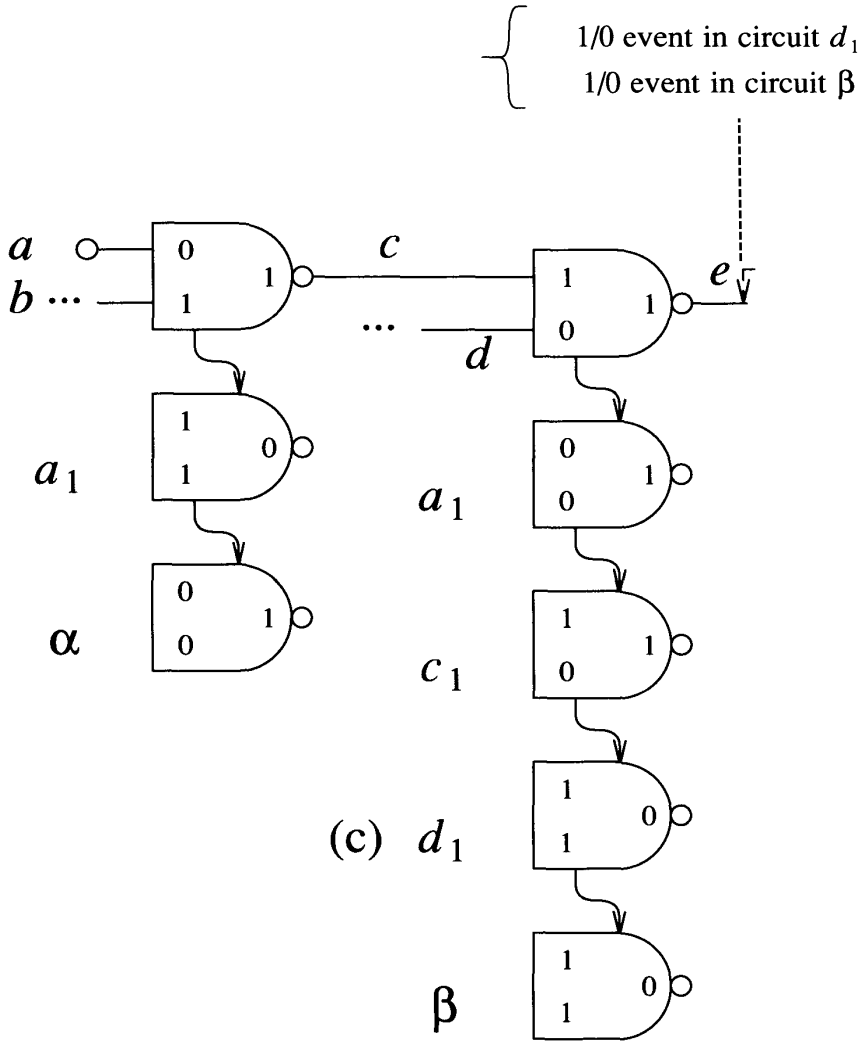
CFS - Example


1/0 good event
a
b
c
d
e
a_1
c_1
d_1
α
β
(a)


0/1 good event
a_1 newly visible
a
b
c
d
e
a_1
c_1
d_1
α
β
(b)

CFS Example – Contd.





Concurrent Simulation

- Individually evaluates elements in both good and faulty circuits
- A line i may change even if i is stable in good circuit (see gate d_1 in previous example)
- A line i in the good circuit and some faulty circuits may also have simultaneous but different events

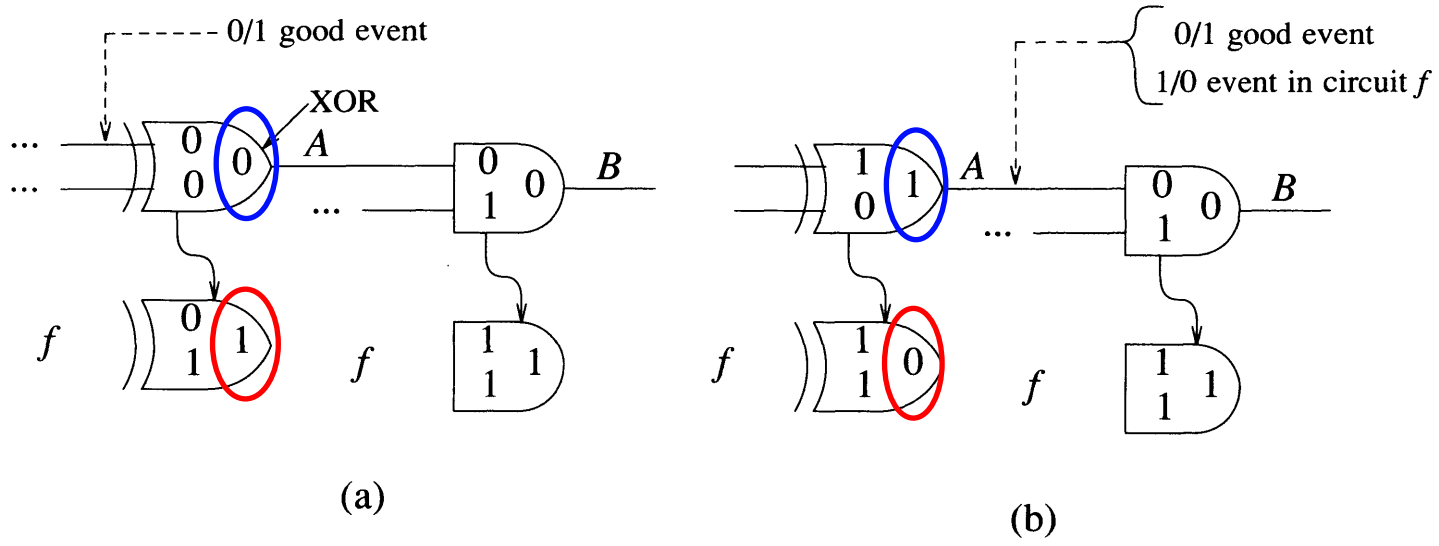


Figure 5.17

Composed Event

- For a given input event on A, we compute the outputs in all copies of A in the fault list
- Let the output list be $L = <(f_0, v'_{f0}), (f_1, v'_{f1}), \ldots (f_n, v'_{fn})>$
- Composed Event:
  - A set of simultaneous events occurring on a line
  - Represented as (i, L)

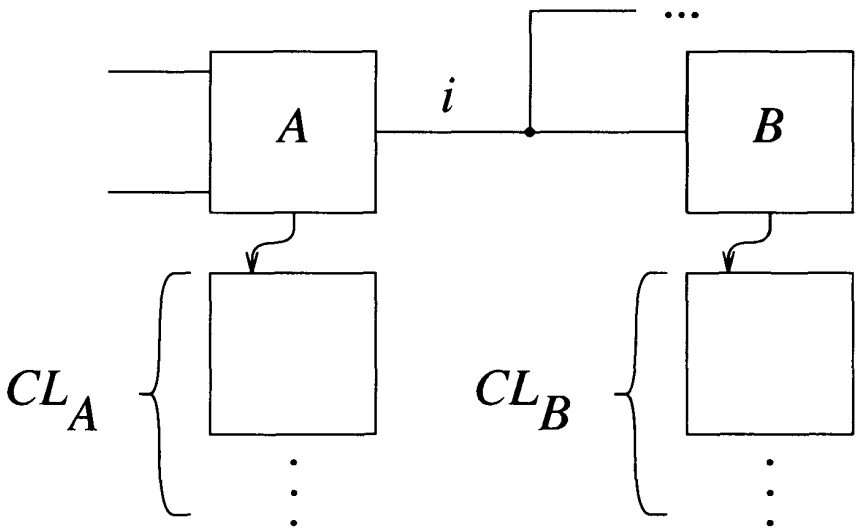


Figure 5.18

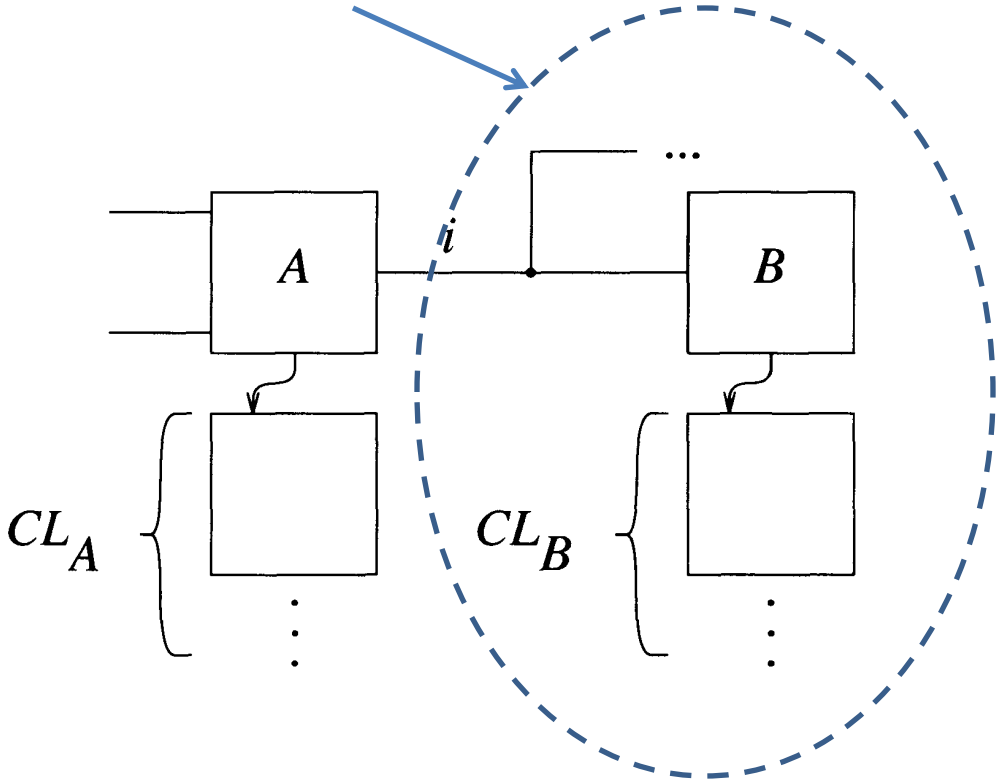
Processing of a composed event (i, L) at element A

```
NV = ∅
if i changes in the good circuit then
  begin
    set i to v' in the good circuit
    for every f ∈ CL_A
      begin
        if f ∈ L then
          begin
            set i to v_f' in circuit f
            if V_Af = V_A then delete f from CL_A
          end
        else /* no event in circuit f */
          if v_f = v then add newly visible fault f to NV
          else if V_Af = V_A then delete f from CL_A
      end
  end
else /* no good event for i */
  for every f ∈ L
    begin
      set i to v_f' in circuit f
      if V_Af = V_A then delete f from CL_A
    end
```

NV: newly visible faults

Processing of element $B_f \in CL_B$

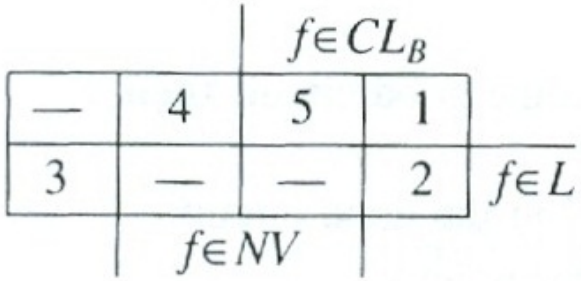
→ After updating the CL_A of source element A, we need to update values and CL_B of every element B on the fanout list of i and evaluate activated elements



	$f \in CL_B$		
—	4	5	1
3	—	—	2
	$f \in NV$		

$f \in L$

Case 1: $f \in CL_B, f \notin L, f \notin NV$



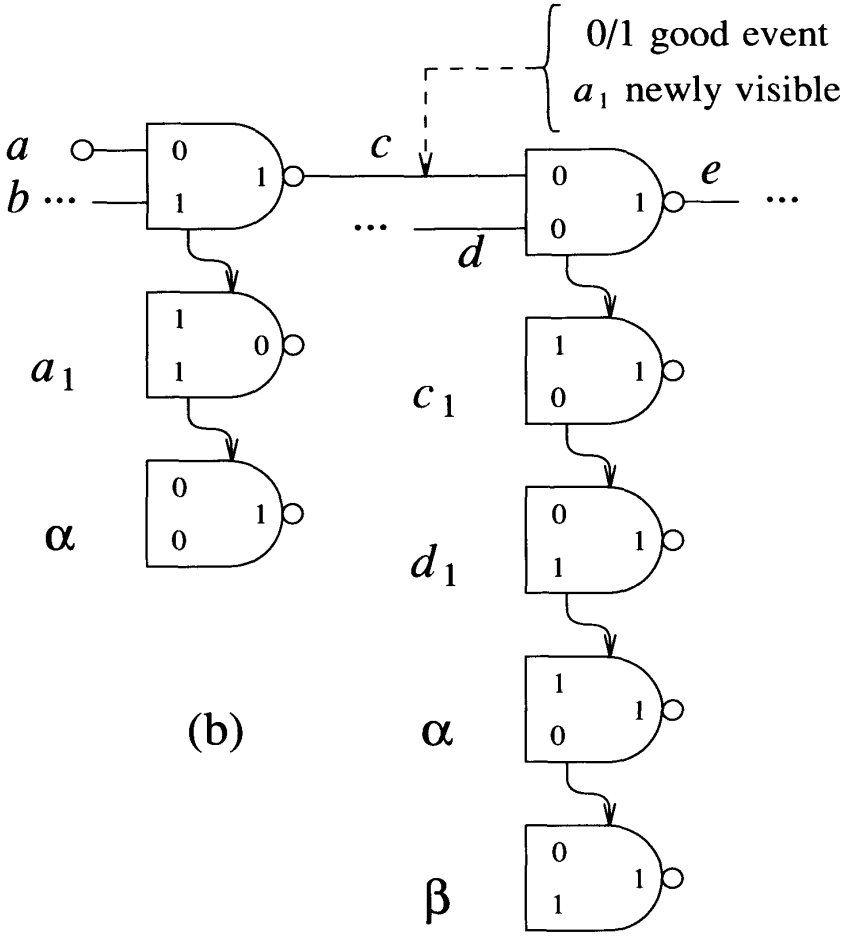
Remark: *B_f exists in CL_B and no independent event on i occurs in N_f*

Action

If good event exists and can propagate in N_f then activate B_f

Example:

Change c 0/1 propagates in d_1 and β but not in c_1 and α



Case 2: $f \in CL_B$, $f \in L$, $f \notin NV$

	$f \in CL_B$		
—	4	5	1
3	—	—	2
	$f \in NV$		

$f \in L$

Remark: *B_f exists in CL_B and an independent event on i occurs in N_f*

Action

Activate B_f

Example:

f in CL_B

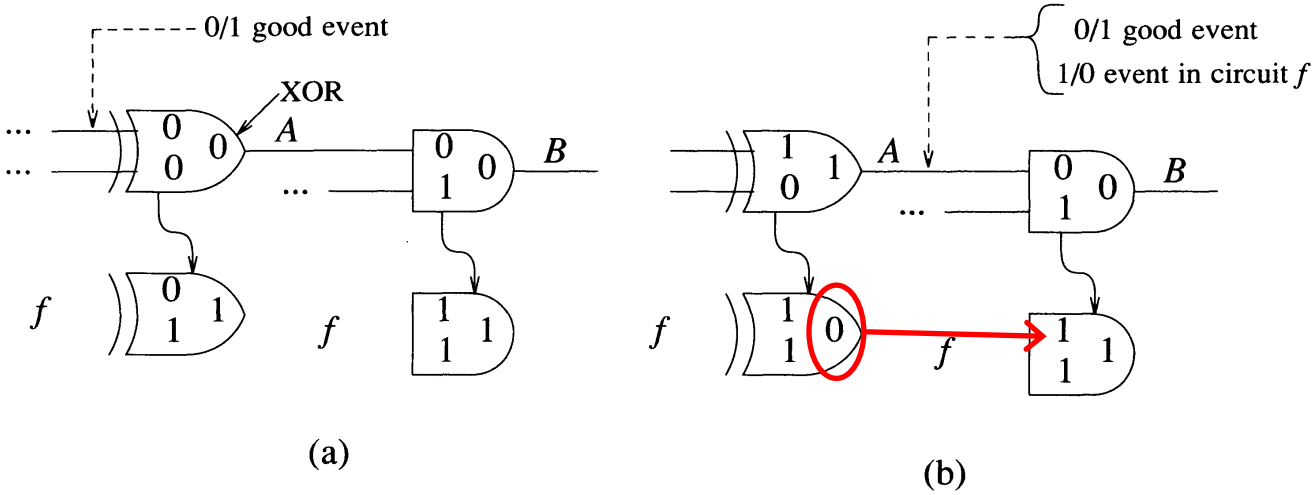
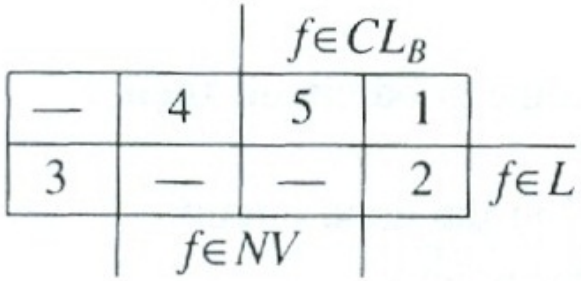


Figure 5.17

Case 3: $f \notin CL_B$, $f \in L$, $f \notin NV$



Remark: A*n independent event on i occurs in N_f* but f does not appear in CL_B

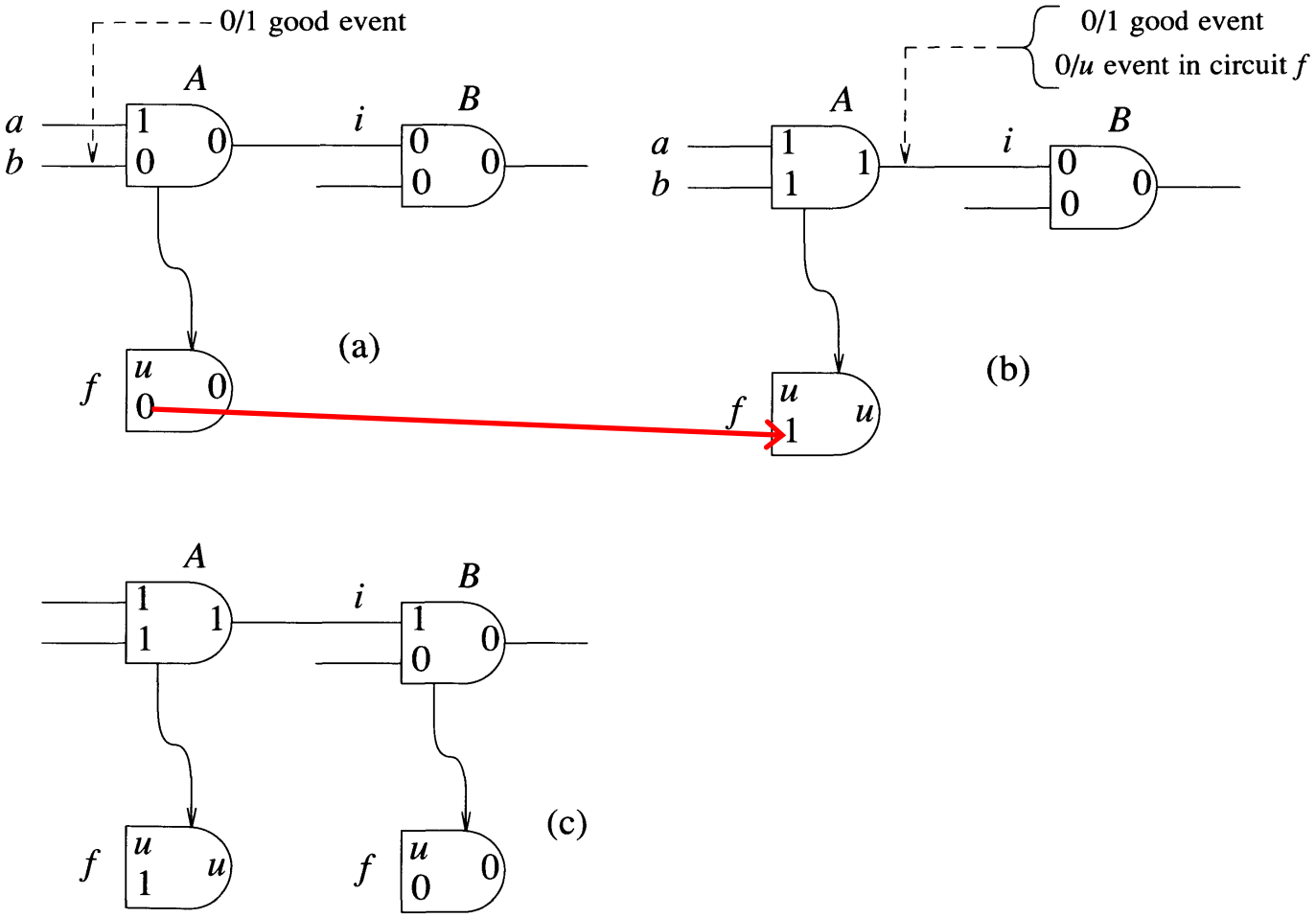
Action: Add an entry for f to CL_B and activate B_f

Case 3: $f \notin CL_B$, $f \in L$, $f \notin NV$

	$f \in CL_B$		
—	4	5	1
3	—	—	2
	$f \in NV$		

$f \in L$

Example:



Case 4: $f \notin CL_B$, $f \notin L$, $f \in NV$

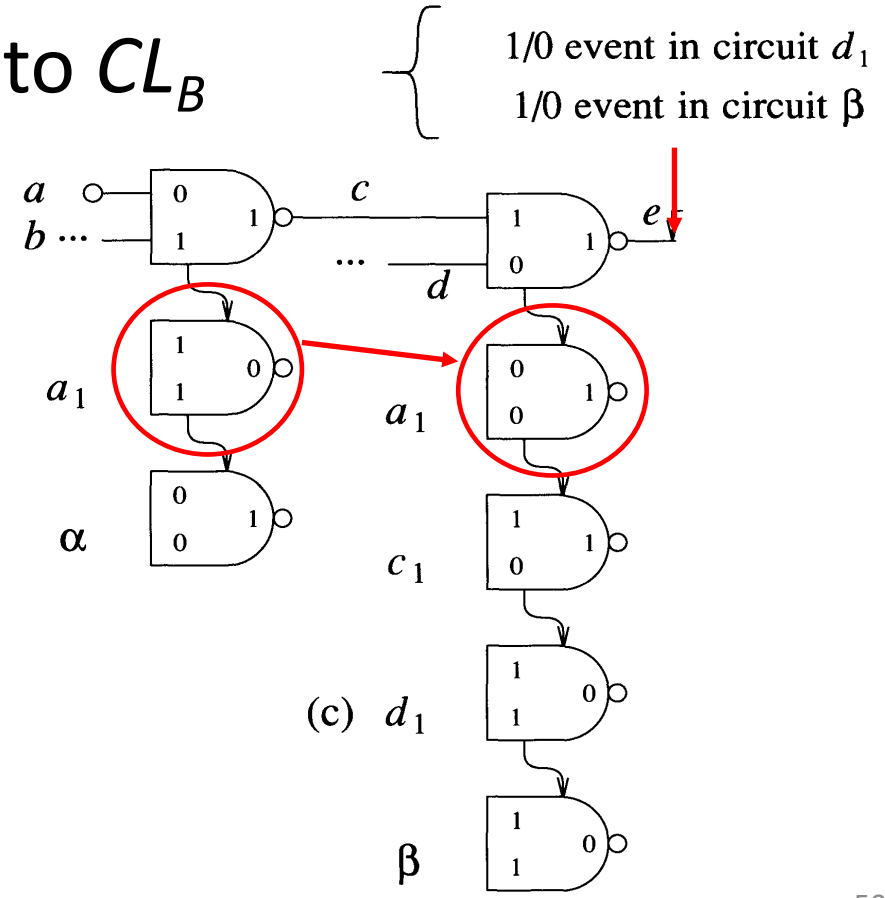
	$f \in CL_B$		
—	4	5	1
3	—	—	2
	$f \in NV$		$f \in L$

Remark: *f is newly visible on line i and does not appear in* CL_B

Action: Add an entry for *f* to CL_B

Example:

Add a_1 to CL_e



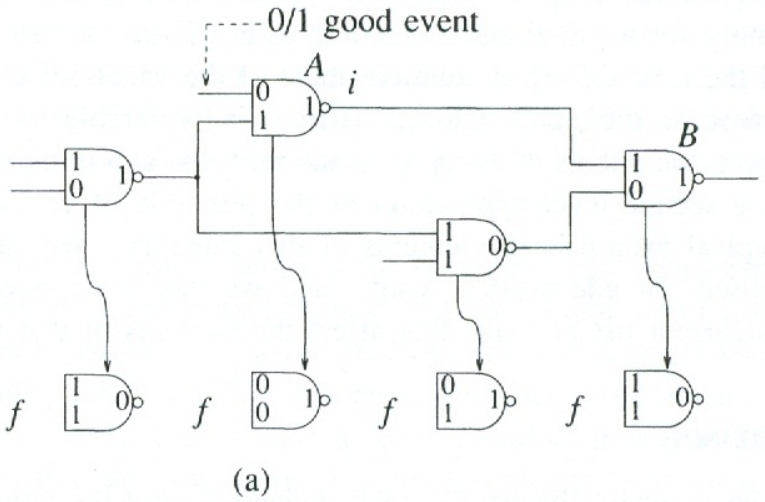
Case 5: $f \in CL_B, f \notin L, f \in NV$

	$f \in CL_B$		
—	4	5	1
3	—	—	2
	$f \in NV$		$f \in L$

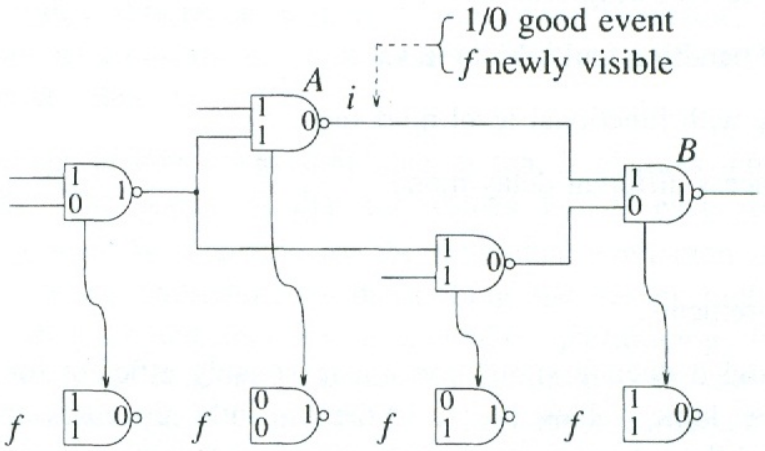
Remark: *f is newly visible on line i but an entry is already present in CL_B*

Action: No Action.

Example: In a comb. circuit this occurs with reconv. fanout



(a)



(b)

Comparison

Criteria	Parallel	Deductive	Concurrent
Multiple Logic Values	Impractical for more than 3 logic values	Impractical for more than 3 logic values	No limit
Functional level Modeling	Partially compatible	Partially compatible	Fully compatible
Different Delay Models	No	No	Yes
Speed*	n^3	n^2	faster?
Storage Reqs.	Medium	Medium	Large

* Comparison for large combinational circuit with *n* gates. No comparison between deductive and concurrent reported.

Backup

Fault Storage

Characteristic Vector

- Fault insertion Bit =1
- Fault deletion Bit = 0
- Union – Bit-wise OR
- Intersection – Bit-wise AND
- Memory Intensive

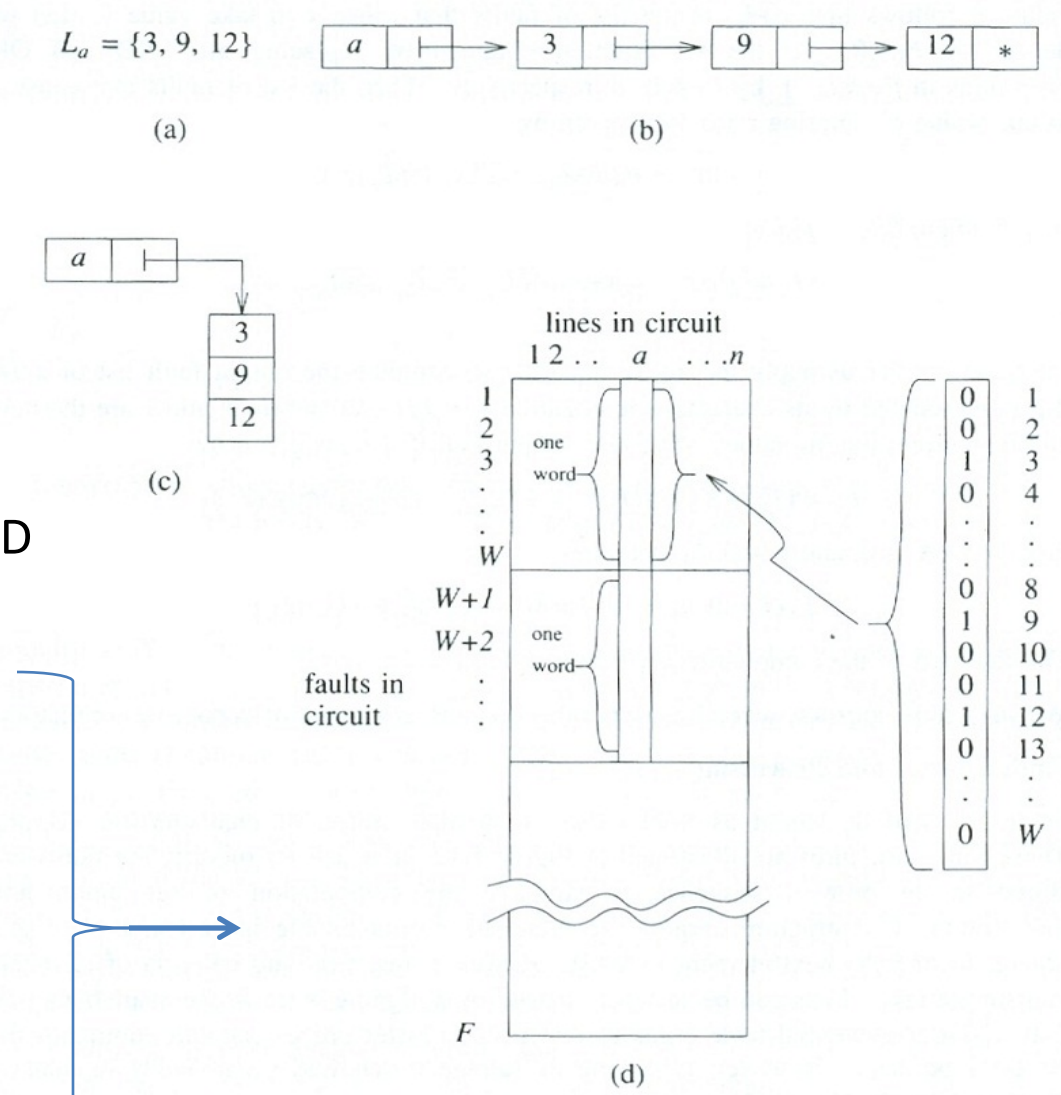


Figure 5.12 Three storage structures for lists (a) Fault list (b) Linked list (c) Sequential table (d) Characteristic vector