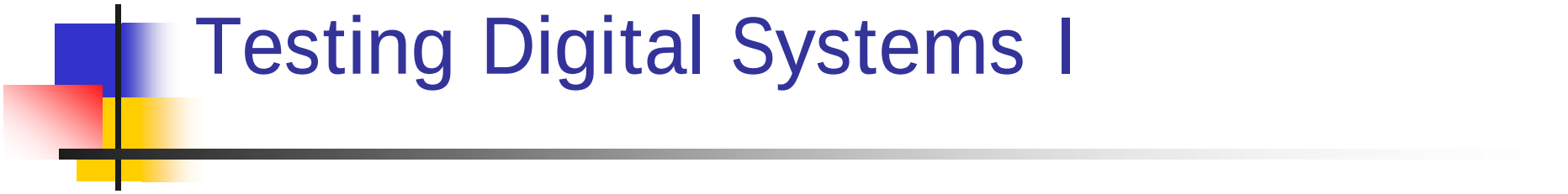
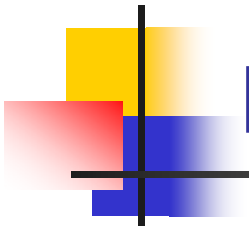


Testing Digital Systems I

Lecture 6:
Fault Simulation

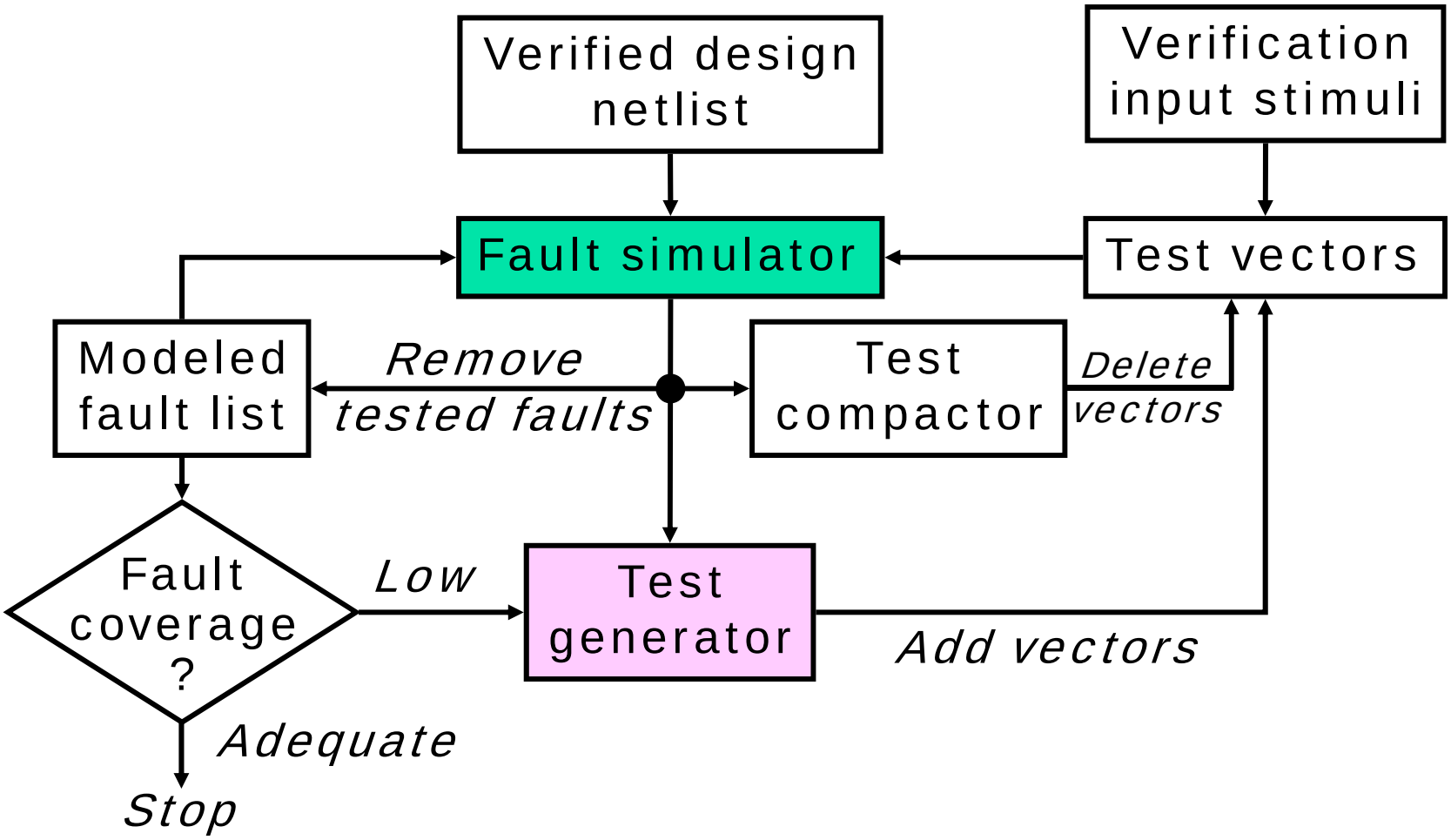
Instructor: M. Tahoori

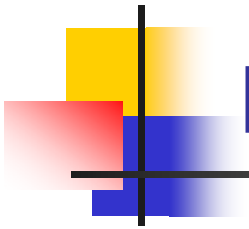


Definition

- Fault Simulator
  - A program that models a design with fault present
- Inputs:
  - A circuit
  - A sequence of test vectors
  - A fault model
    - Usually single-stuck faults
    - Sometimes multiple-stuck, bridging faults,…
- Determines
  - Fault coverage (fault grading)
  - Set of undetected faults (Areas of Low Fault Coverage)
  - Generates fault dictionary (Fault diagnosis)
  - Test compaction

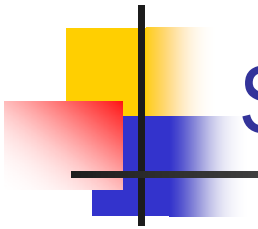
Fault simulator in VLSI Design





Fault Simulation Algorithms

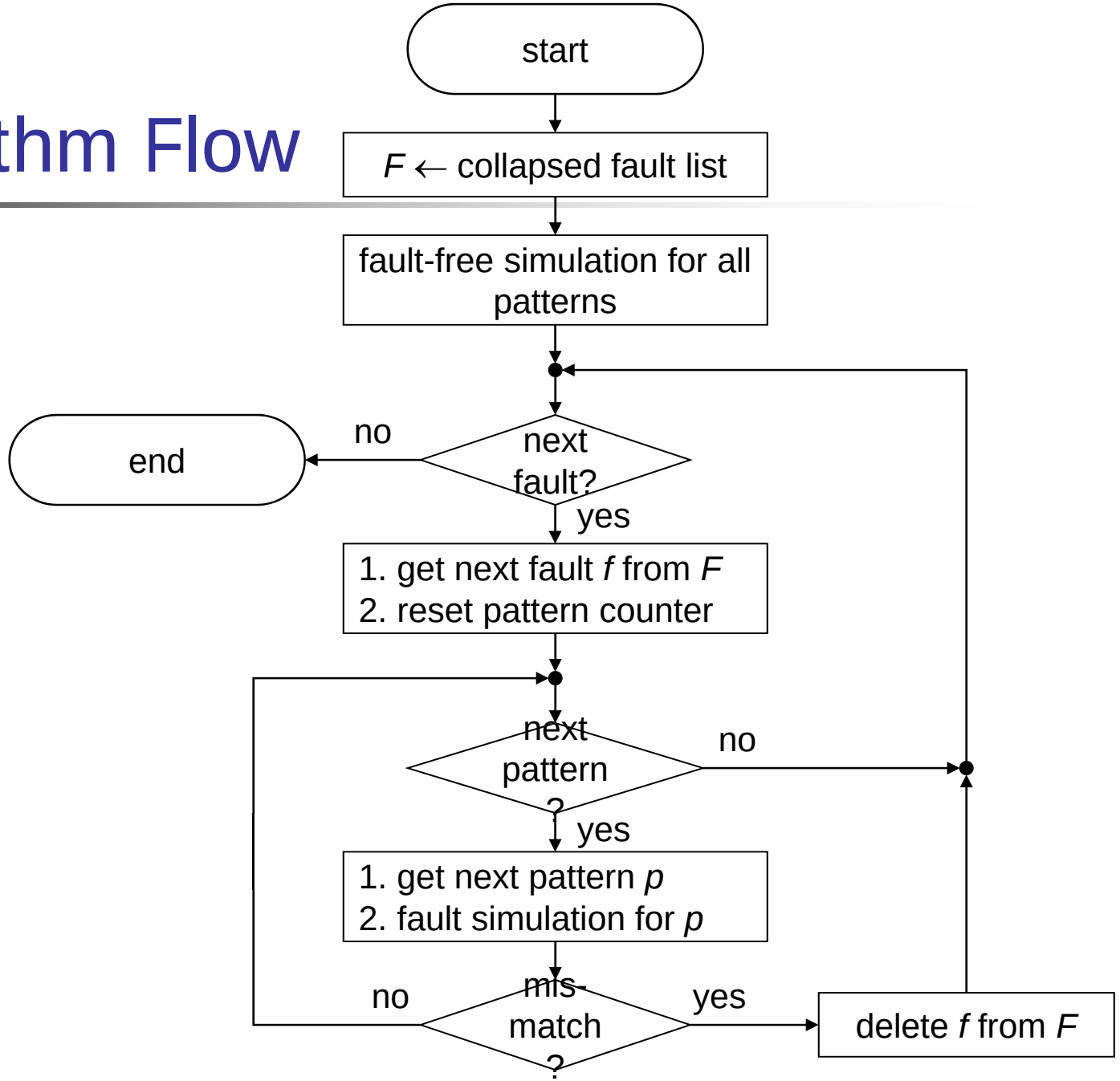
- Serial
- Parallel
- Deductive
- Concurrent



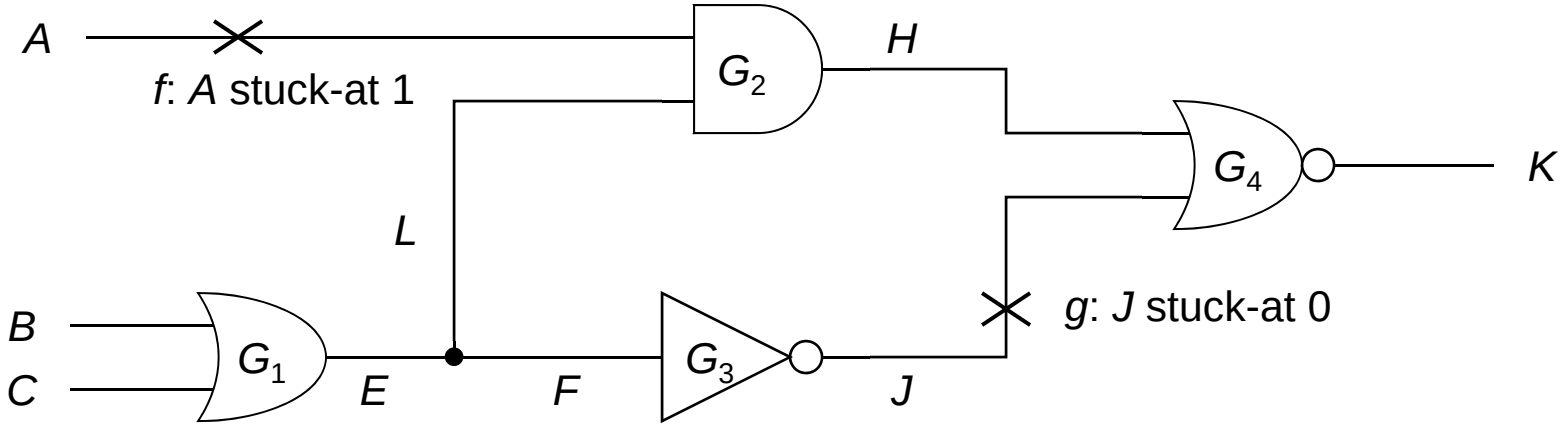
Serial Fault Simulation

- First, perform fault-free logic simulation on the original circuit
  - Good (fault-free) response
- For each fault, perform fault injection and logic simulation
  - Modify Circuit by Introducing Fault i
    - Obtain faulty circuit N_i
  - Simulate modified netlist, vector by vector, comparing responses with good responses

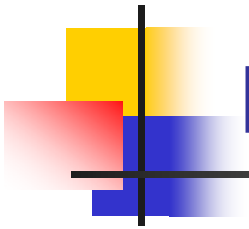
Algorithm Flow



Example

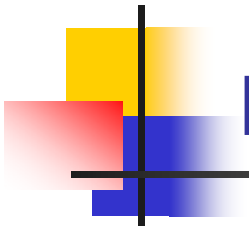


Pat. #	Input			Internal					Output		
	A	*B*	*C*	*E*	*F*	*L*	*J*	*H*	K_{good}	K_f	K_g
P1	0	1	0	1	1	1	0	0	1	**0**	1
P2	0	0	1	1	1	1	0	0	1	**0**	1
P3	1	0	0	0	0	0	1	0	0	0	**1**



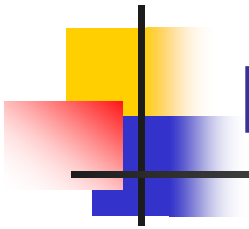
Fault Dropping

- Halting simulation of the detected fault
- Example
  - Suppose we are to simulate P_1, P_2, P_3 in order
  - Fault f is detected by P_1
  - Do not simulate f for P_2, P_3
- For fault grading
  - Most faults are detected after relatively few test patterns have been applied
- For fault diagnosis
  - Avoided to obtain the entire fault simulation results



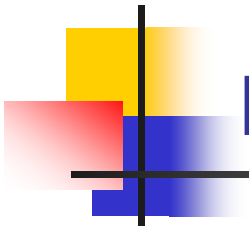
Pro and Con

- Advantages:
  - Easy to implement
  - Compatible with hardware accelerator
  - Ability to handle a wide range of fault models
    - (stuck-at, delay, Br, …)
  - Very fast combinational simulation
    - Simulate many input patterns in parallel
- Disadvantage
  - Many simulation runs required
    - CPU time prohibitive for VLSI circuits



Parallel Fault Simulation

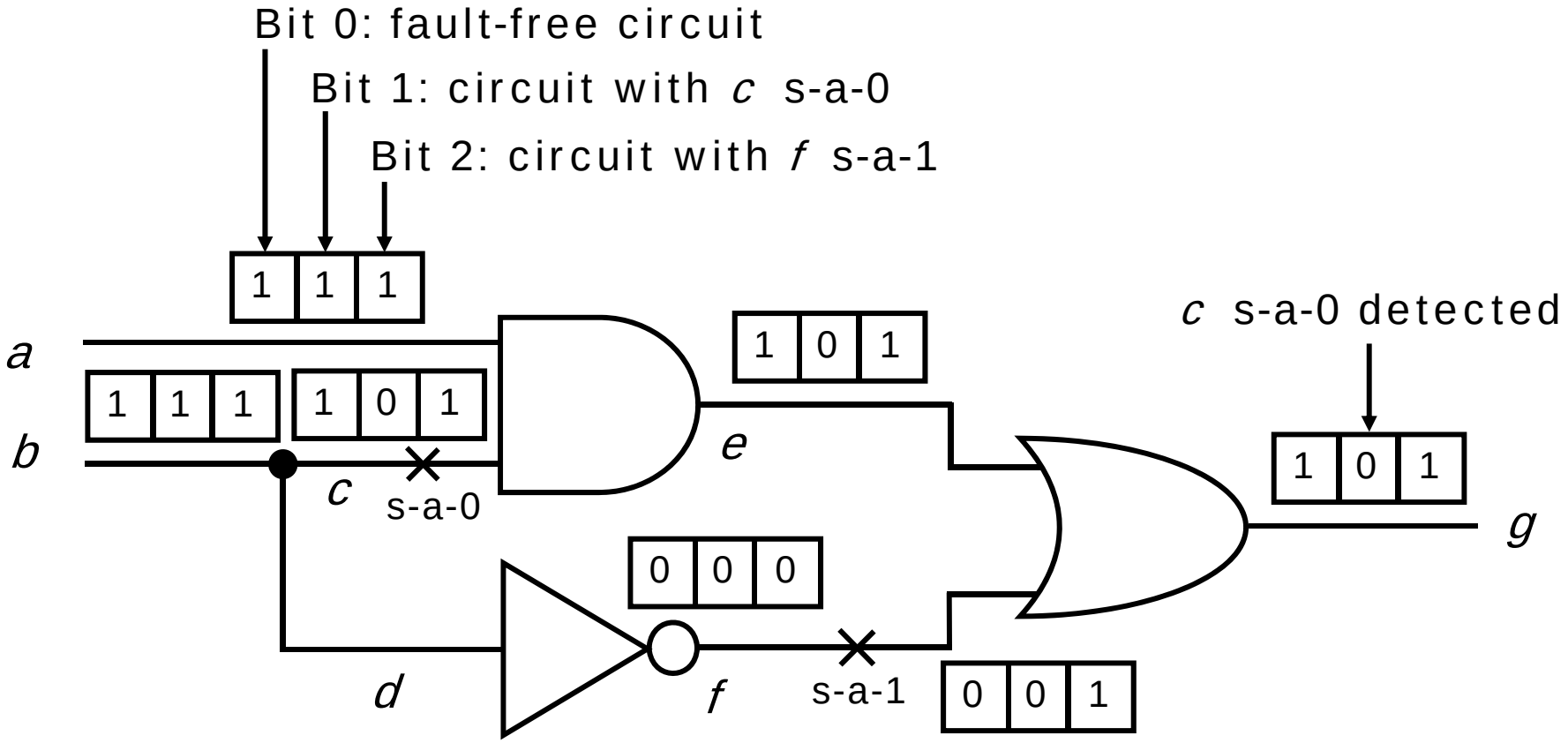
- Compiled-code method; best with two-states (0,1)
- Exploits inherent bit-parallelism of logic operations on computer words
- Each Signal Line is Represented by a Vector
  - Bit 0 of Computer Word Represents Good Circuit
    - Value of Same Signal in Good Circuit
  - Remaining Bits i of Computer Word represent
    - Values of Same Signal in Faulty Circuits Ni
- Multi-pass simulation:
  - Each pass simulates w-1 faults, where w is the machine word length
- Speed up over serial method ~ w-1



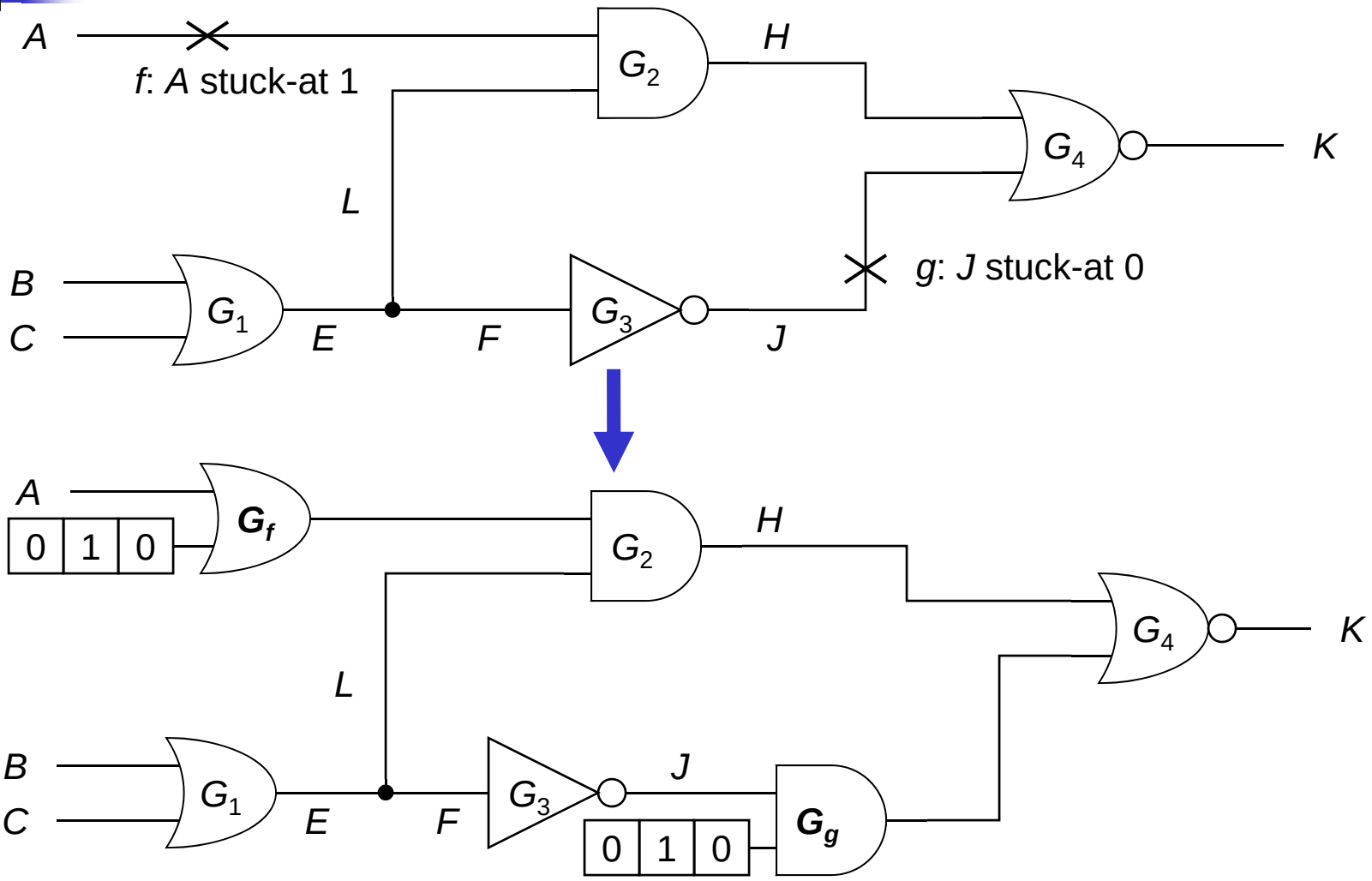
Parallel Fault Simulation

- How about 3-valued signals?
  - Use two vectors for each signal line
    - Signal Value 0 1 X
    - Vector V^1 bit value 0 1 0
    - Vector V^2 bit value 0 1 1
  - AND Gate with inputs A, B; output C
    - $C^1 = A^1B^1$; $C^2 = A^2B^2$;
  - Inverter with inputs A; output B
    - $B^1 = (A^2)'$; $B^2 = (A^1)'$
- Fault Insertion Masks
  - Presence of fault on each line
  - The value of faulty line

Parallel Fault Simulation

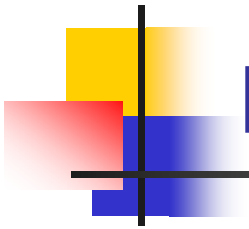


Fault Injection



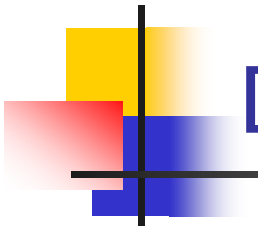
Example

Pat #		Input				Internal						Output
		A	A_f	***B***	***C***	***E***	***F***	***L***	***J***	J_g	***H***	***K***
P_1	FF	0	0	1	0	1	1	1	0	0	0	1
	f	**0**	**1**	1	0	1	1	1	0	0	**1**	**0**
	g	0	0	1	0	1	1	1	0	0	0	1
P_2	FF	0	0	0	1	1	1	1	0	0	0	1
	f	0	**1**	0	1	1	1	1	0	0	**1**	**0**
	g	0	0	0	1	1	1	1	0	0	0	1
P_3	FF	1	1	0	0	0	0	0	1	1	0	0
	f	1	1	0	0	0	0	0	1	1	0	0
	g	1	1	0	0	0	0	0	1	**0**	0	**1**



Pro and Con

- Advantages
  - A large number of faults are detected by each pattern when simulating the beginning of test sequence
- Disadvantages
  - Only applicable to the unit or zero delay models
  - Faults cannot be dropped unless all (w-1) faults are detected



Deductive Fault Simulation

- Only good circuit is explicitly simulated
  - Deduce from good signals all detectable faults
- Each circuit line i has list Li of faults
  - Each fault causes line-i error for current input state
    - Lists propagated from primary inputs to outputs
    - Lists updated for each input state change
- Event driven (selective trace)
  - logic event — signal value changes
  - fault event — fault list changes
- Recompute list wherever either event occurs

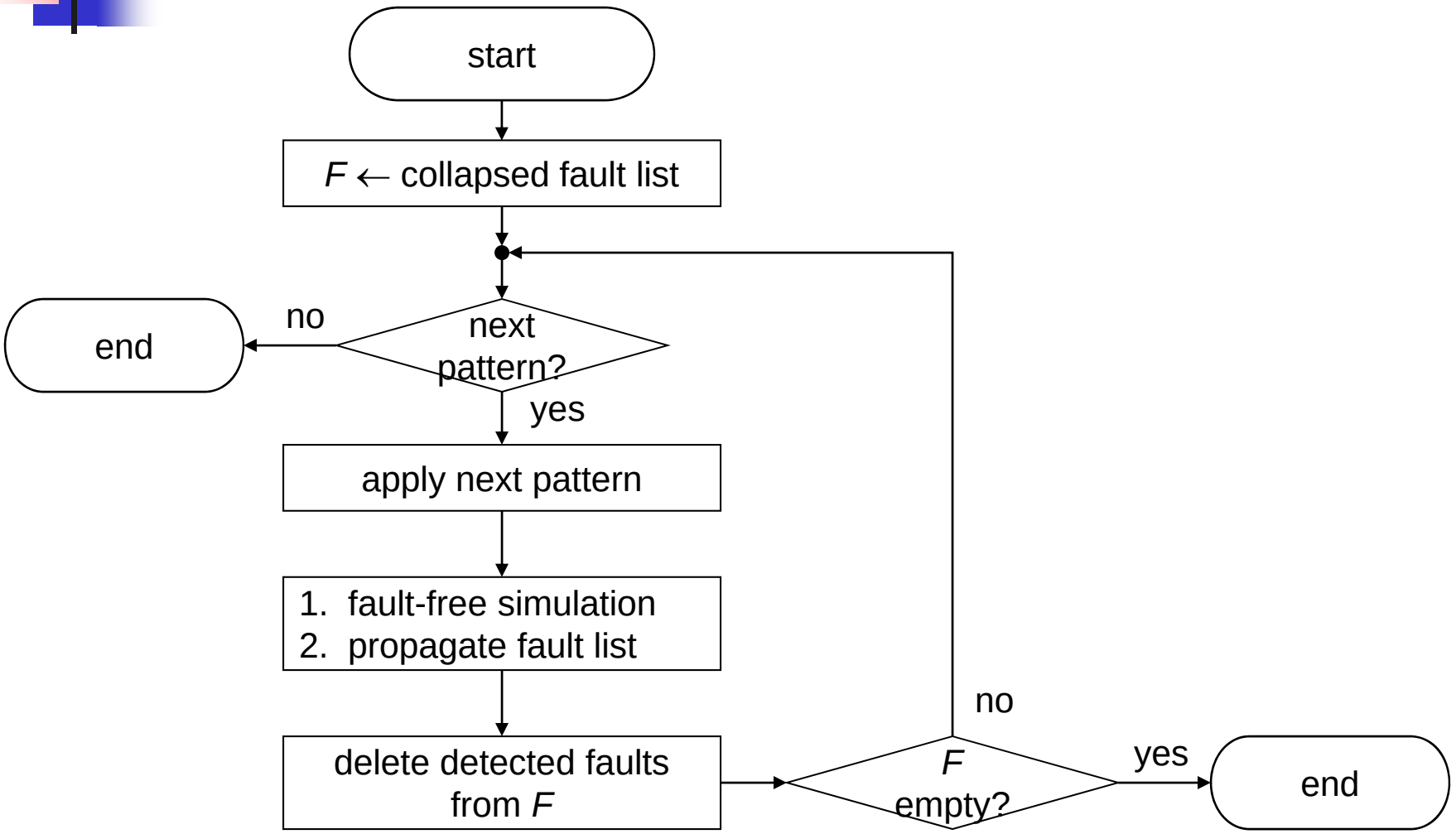
+ One pass for each input pattern (in principle)

- Set-theoretic rules difficult to derive for non-Boolean gates
- Gate delays are difficult to use
- Memory Space Needed Cannot be Predicted

Fault List Propagation Rules

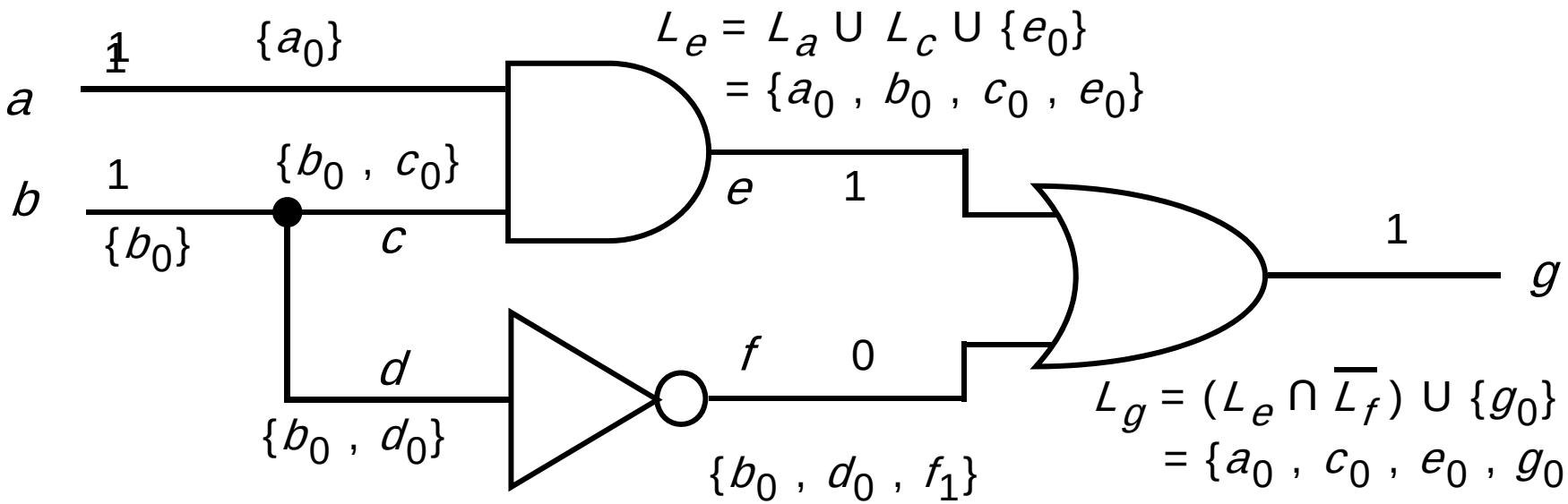
Gate type	inputs		Output	Output fault list
	a	b	c	L_c
AND	0	0	0	$[L_a \cap L_b] \cup c_1$
	0	1	0	$[L_a \cap L_b'] \cup c_1$
	1	0	0	$[L_a' \cap L_b] \cup c_1$
	1	1	1	$[L_a \cup L_b] \cup c_0$
OR	0	0	0	$[L_a \cup L_b] \cup c_1$
	0	1	1	$[L_a' \cap L_b] \cup c_0$
	1	0	1	$[L_a \cap L_b'] \cup c_0$
	1	1	1	$[L_a \cap L_b] \cup c_0$
NOT	0	-	1	$L_a \cup c_0$
	1	-	0	$L_a \cup c_1$

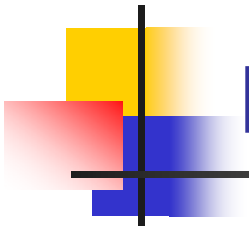
Algorithm Flow



Deductive Fault Simulation

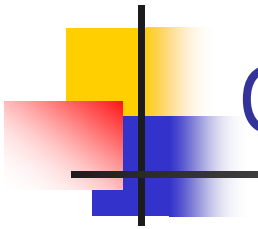
Notation: L_k is fault list for line k
k_n is s-a-n fault on line k





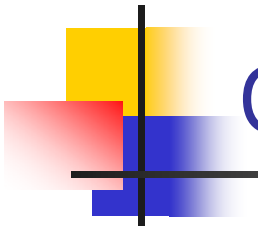
Pro and Con

- Advantages
  - Very efficient
  - Simulate all faults in one pass
- Disadvantages
  - Not easy to handle unknowns
  - Only for zero-delay timing model
  - Potential memory management problem



Concurrent Fault Simulation

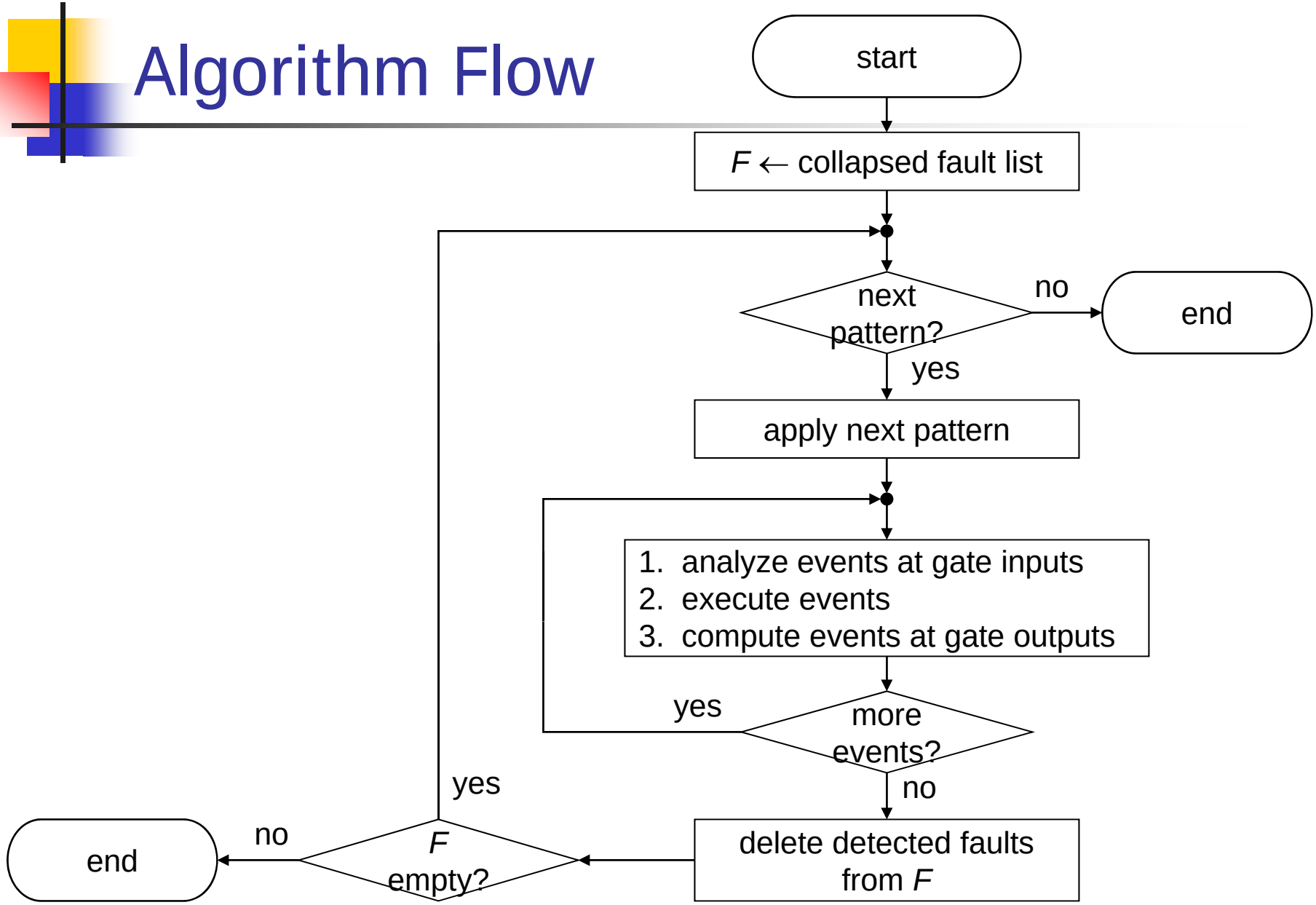
- Simulate only differential parts of whole circuit
- Event-driven simulation with fault-free and faulty circuits simulated altogether
- Concurrent fault list for each gate
  - Consist of a set of bad gates
    - Fault index & associated gate I/O values
  - Initially only contains local faults
  - Fault propagate from previous stage



Good Event and Bad Event

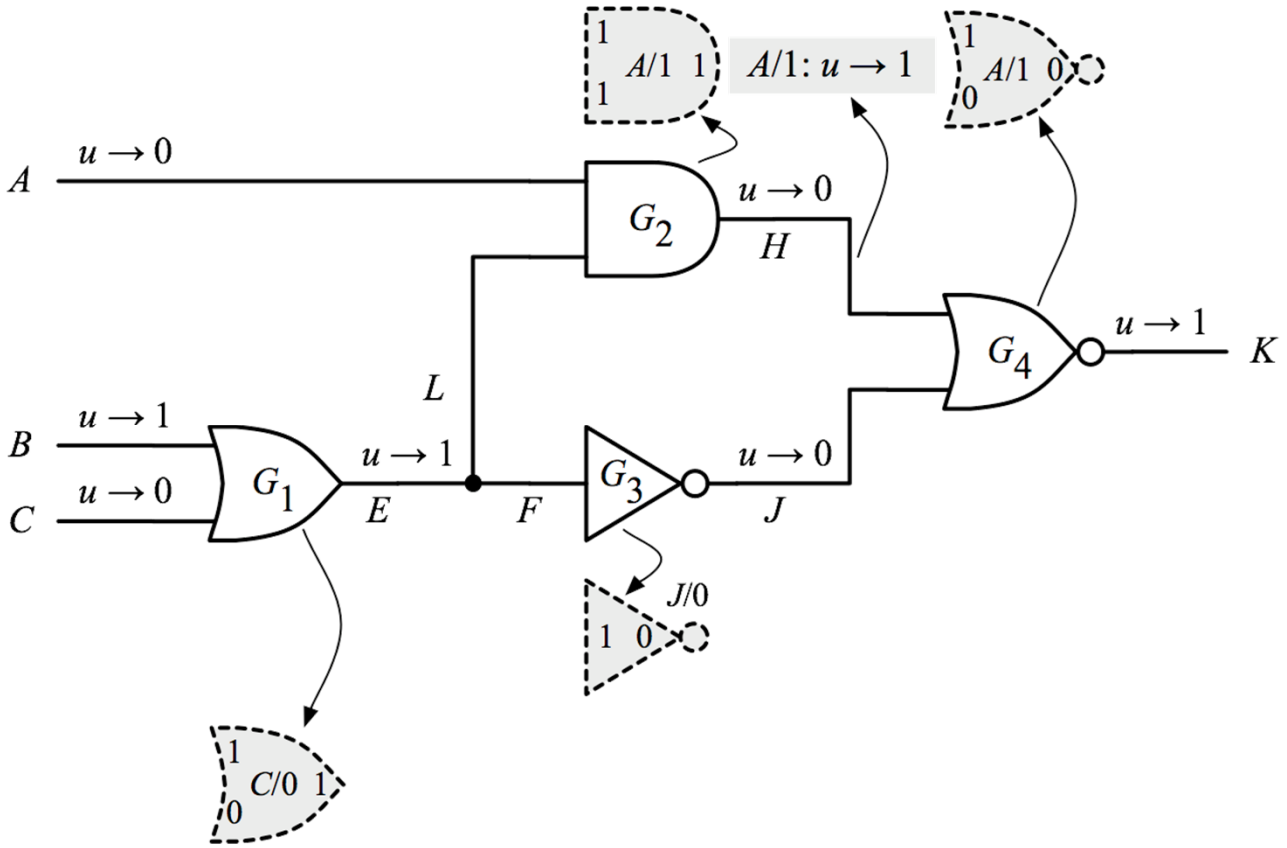
- Good event
  - Events that happen in good circuit
  - Affect both good gates and bad gates
- Bad event
  - Events that occur in the faulty circuit of corresponding fault
  - Affect only bad gates
- Diverge
  - Addition of new bad gates
- Converge
  - Removal of bad gates whose I/O signals are the same as corresponding good gates

Algorithm Flow



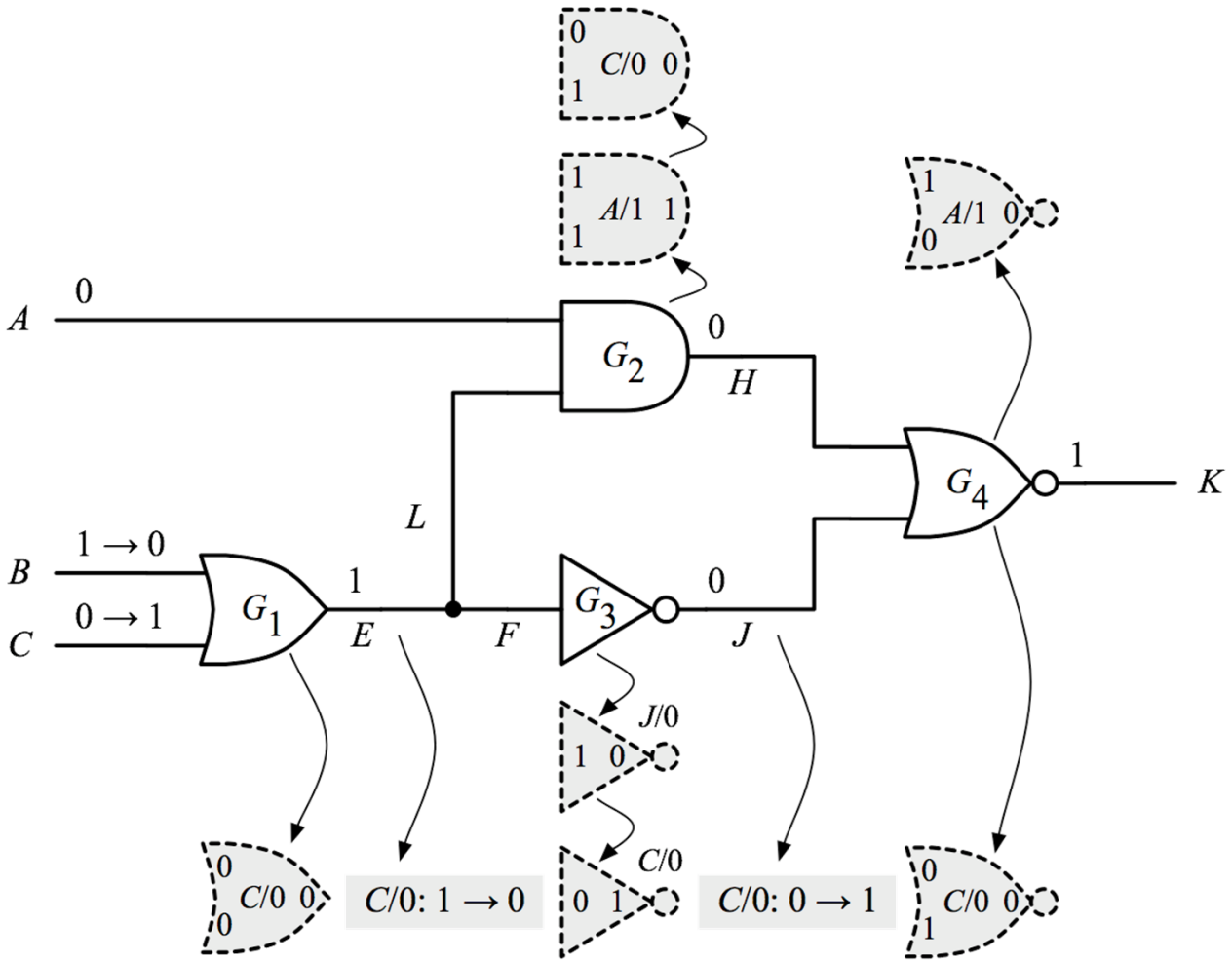
Example

- P_1


A
u → 0
B
u → 1
C
u → 0
G1
G2
G3
G4
L
E
F
H
J
K
u → 1
u → 0
u → 1
u → 0
u → 1
A/1 1
1
1
A/1: u → 1
A/1 0
1
0
J/0
1 0
C/0 1
1
0

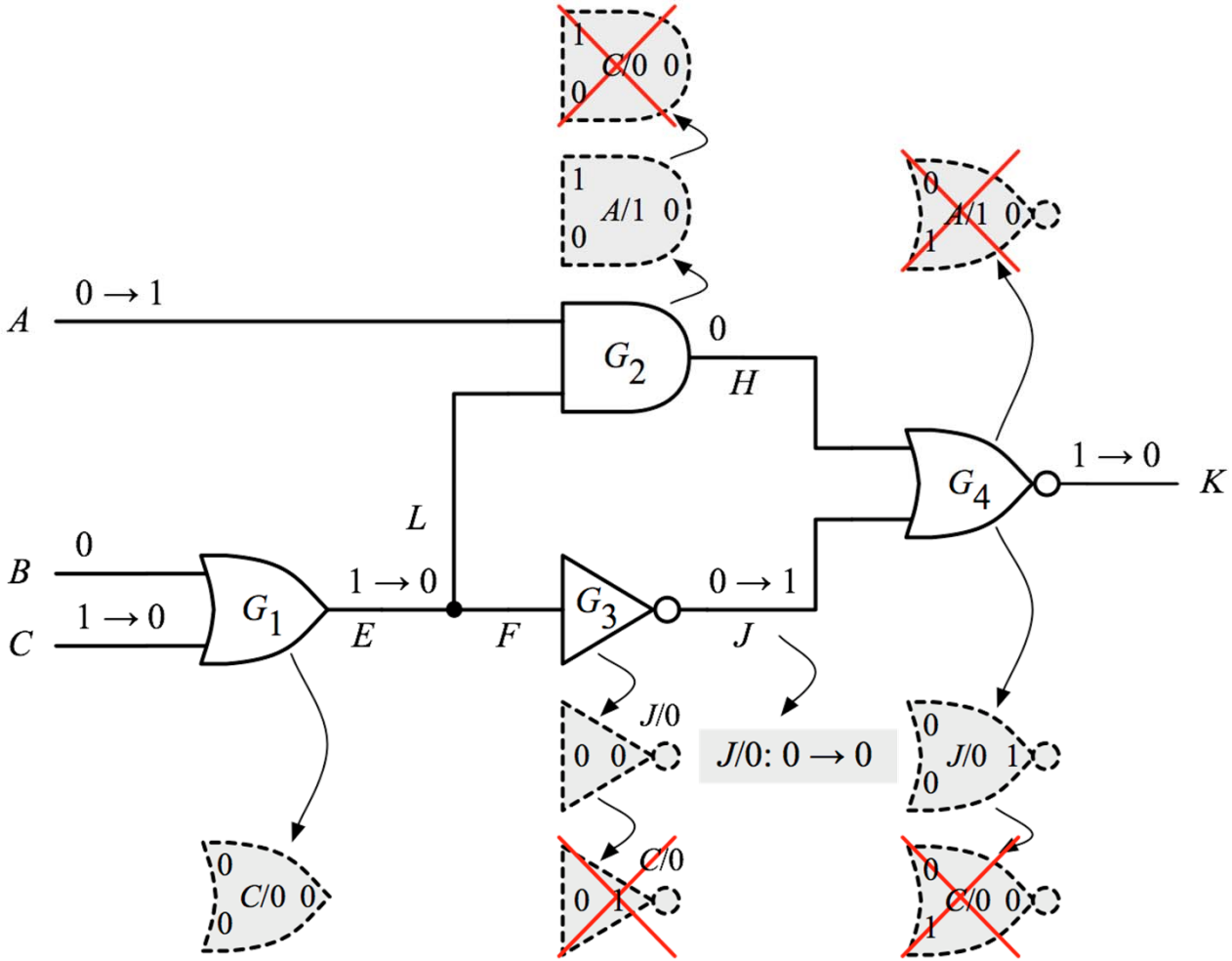
Example (cont'd)

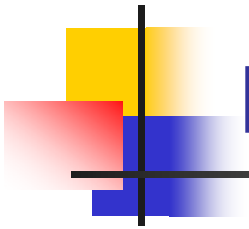
- P_2



Example (cont'd)

- P_3





Pro and Con

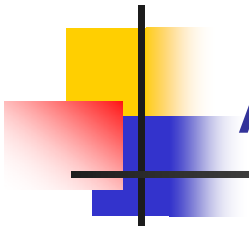
- Advantages
  - Efficient
  - Faults can be simulated in any modeling style or detail supported in true-value simulation (offers most flexibility.)
  - Faster than other methods
- Disadvantages
  - Potential memory problem
    - Size of the concurrent fault list changes at run time

Comparison of Fault Simulation Techniques

- Speed
  - Serial fault simulation: slowest
  - Parallel fault simulation: O(n^3), n: num of gates
  - Deductive fault simulation: O(n^2)
  - Concurrent fault is faster than deductive fault simulation
- Memory usage
  - Serial fault simulation, parallel fault simulation: no problem
  - Deductive fault simulation: dynamic allocate memory and hard to predict size
  - Concurrent fault simulation: more severe than deductive fault simulation

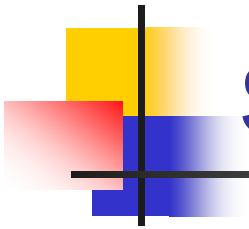
Comparison of Fault Simulation Techniques

- Multi-valued fault simulation to handle unknown (X) and/or high-impedance (Z)
  - Serial fault simulation, concurrent fault simulation:
    - easy to handle
  - Parallel fault simulation: difficult
- Delay and functional modeling capability
  - Serial fault simulation: no problem
  - Parallel fault simulation, deductive fault simulation: not capable
  - Concurrent fault simulation: capable



Alternative to Fault Simulation

- Toggle Coverage
- Fault Sampling
- Critical Path Tracing
- Statistical Fault Analysis



Summary

- Fault simulation is very important for
  - ATPG
  - Diagnosis
  - Fault grading
- Popular techniques
  - Serial, Parallel, Deductive, Concurrent
- Requirements for fault simulation
  - Fast speed, efficient memory usage, modeling functional blocks, sequential circuits