VLSI Test Technology and Reliability (ET4076)

Lecture 4(part 2)

Testability Measurements

(Chapter 6)

Said Hamdioui

Computer Engineering Lab
Delft University of Technology
2009-2010

Previous lecture

- What is the difference between Functional and Structural testing? Give an example
- What is a defect? What is a fault model?
- What are the common fault models?
- How are the following concept used for test generation
  - Single-stuck at fault
  - Fault equivalence
  - Fault dominance and checkpoint theorem
- What are the transistor faults?

Learning aims of today

- Describe concepts like simulation, simulation for design, simulation for test
- Describe the different models used for circuit simulation (gate level, timing, signals, etc)
- Distinguish the two different methods for logic simulation (concept, advantages, …)
- Distinguish the four different methods for fault simulation (concept, advantages, …)
- Describe alternatives for fault simulation

Logic Simulation & Fault Simulation

Logic Simulation

- Definition of simulation
- Simulation for design verification
- Simulation models
- Logic simulation
  - Compiled-code simulation
  - Event-driven simulation
- Summary

Fault simulation

- Problem and motivation
- Simulation for Test
- Fault simulation algorithms
  - Serial
  - Parallel
  - Deductive
  - Concurrent
- Alternatives to fault simulation
  - Random Fault Sampling
- Summary

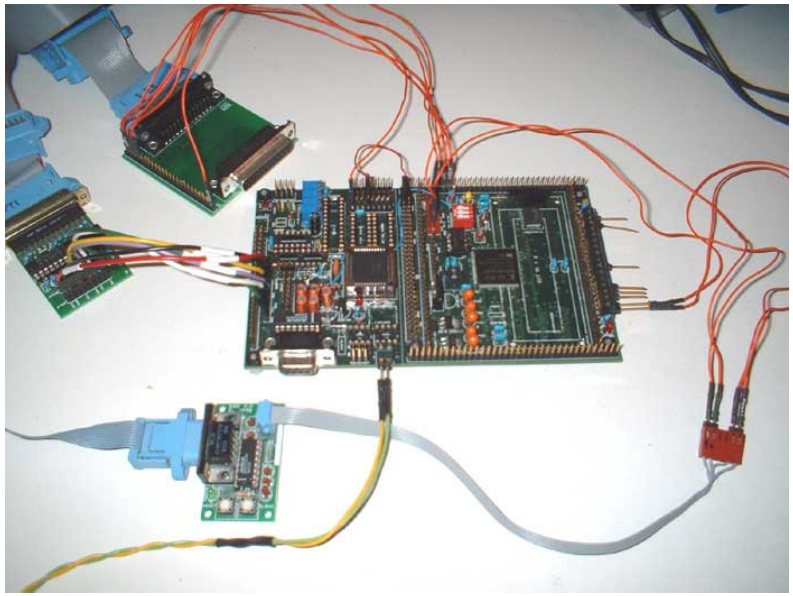
Logic Simulation

- Definition of simulation
- Simulation for design verification
- Simulation models
- Logic simulation
  - Compiled-code simulation
  - Event-driven simulation
- Summary

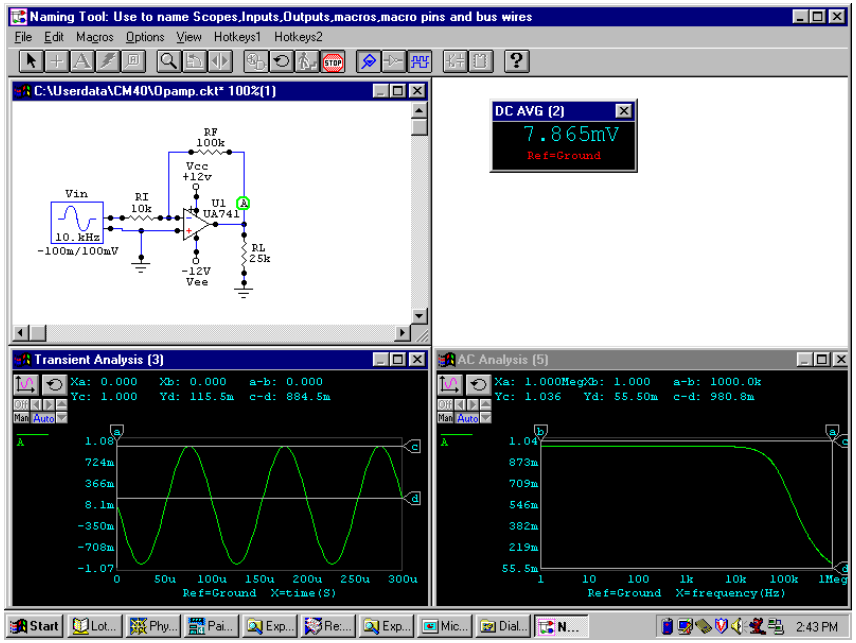
Definition of simulation

- Definition: **Simulation** refers to modeling of a design, its function and performance.
- In electronic design, two purposes:
  - Verify the correctness of the **design**
  - Verify the **tests**

Hardware simulator (emulator)



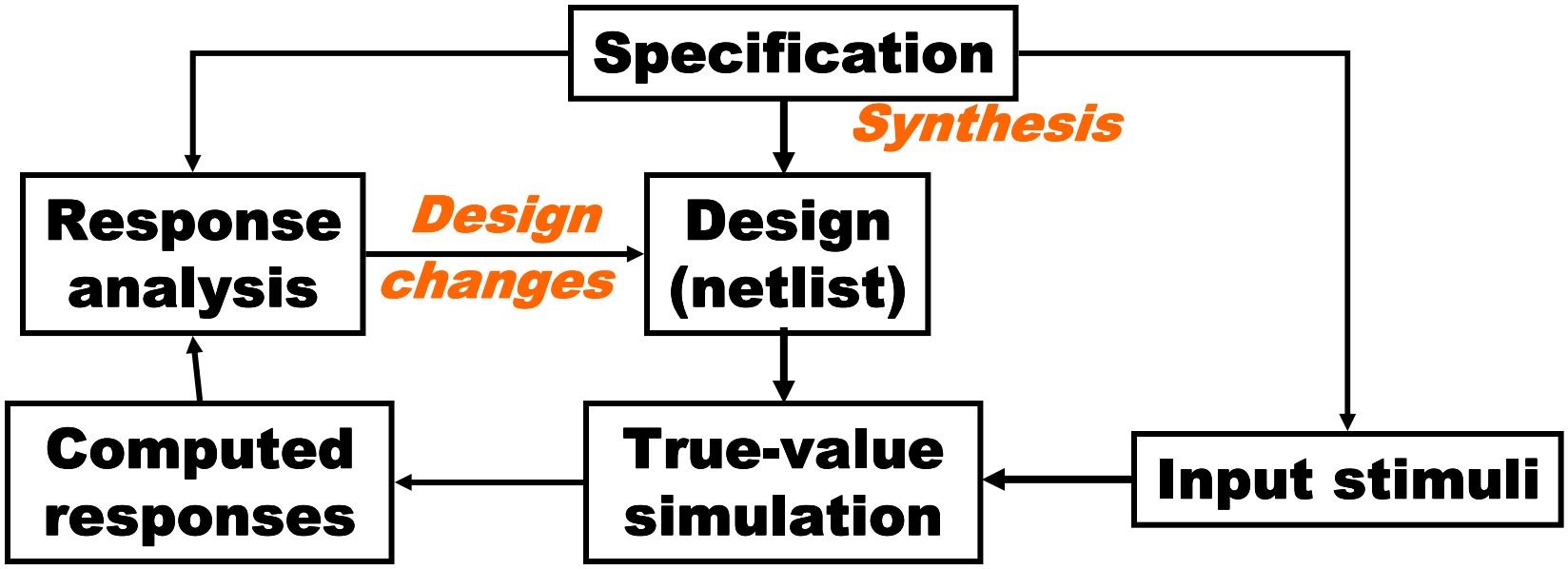
Software simulator



Definition of simulation

- Simulation for **design verification**:
  - Validate assumptions
  - Verify logic
  - Verify performance (timing)
  - Verify the specs
  - Identify **design errors**
- Simulation at different levels:
  - Behavioral/RTL/functional
  - Logic
  - Switch level
  - Timing
  - Circuit/layout

Simulation for design verification



- Advantages:
  - Details of circuit behavior can be simulation
  - Use of hierarchy (e.g., vhdl/C before netlist)
- Weaknesses
  - Incomplete: Dependency on design's heuristics
  - E.g., adder for two 32-bit integer: 2^{64} input vector (impractical)

Simulation models

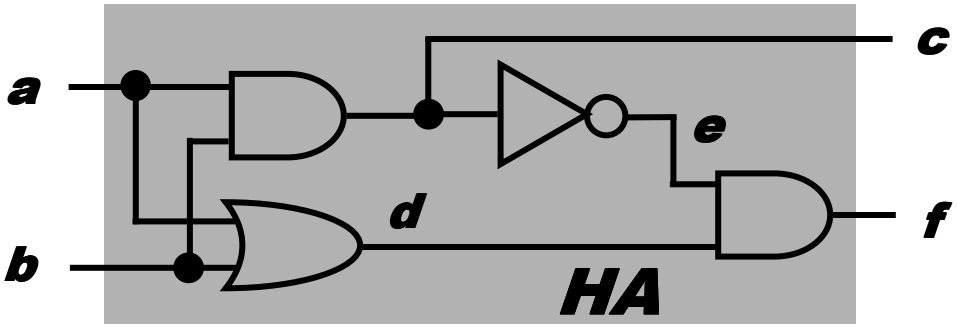
- Modules, blocks or components described by
  - Input/output (I/O) function
  - Delays associated with I/O signals
  - Examples: binary adder, Boolean gates, FET, resistors and capacitors
- Interconnects represent
  - Ideal signal carriers, or
  - Ideal electrical conductors
- **Netlist**: a format (or language) that describes a design as an interconnection of modules. Netlist may use hierarchy.

Simulation models

Modeling level	Circuit description	Signal values	Timing	Application
Function, behavior, RTL	Programming language-like HDL	0, 1	Clock boundary	Architectural and functional verification
Logic	Connectivity of Boolean gates, flip-flops and transistors	0, 1, X and Z	Zero-delay unit-delay, multiple-delay	Logic verification and test
Switch	Transistor size and connectivity, node capacitances	0, 1 and X	Zero-delay	Logic verification
Timing	Transistor technology data, connectivity, node capacitances	Analog voltage	Fine-grain timing	Timing verification
Circuit	Tech. Data, active/ passive component connectivity	Analog voltage, current	Continuous time	Digital timing and analog circuit verification

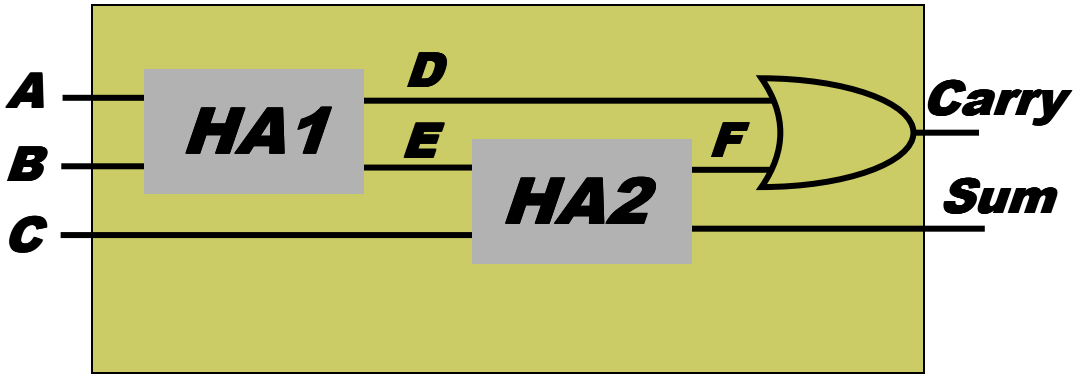
Simulation models….. Hierarchy connectivity

Half Adder



***HA**;*
***inputs**: a, b;*
***outputs**: c, f;*
AND: A1, (a, b), (c);
AND: A2, (d, e), (f);
OR: O1, (a, b), (d);
NOT: N1, (c), (e);

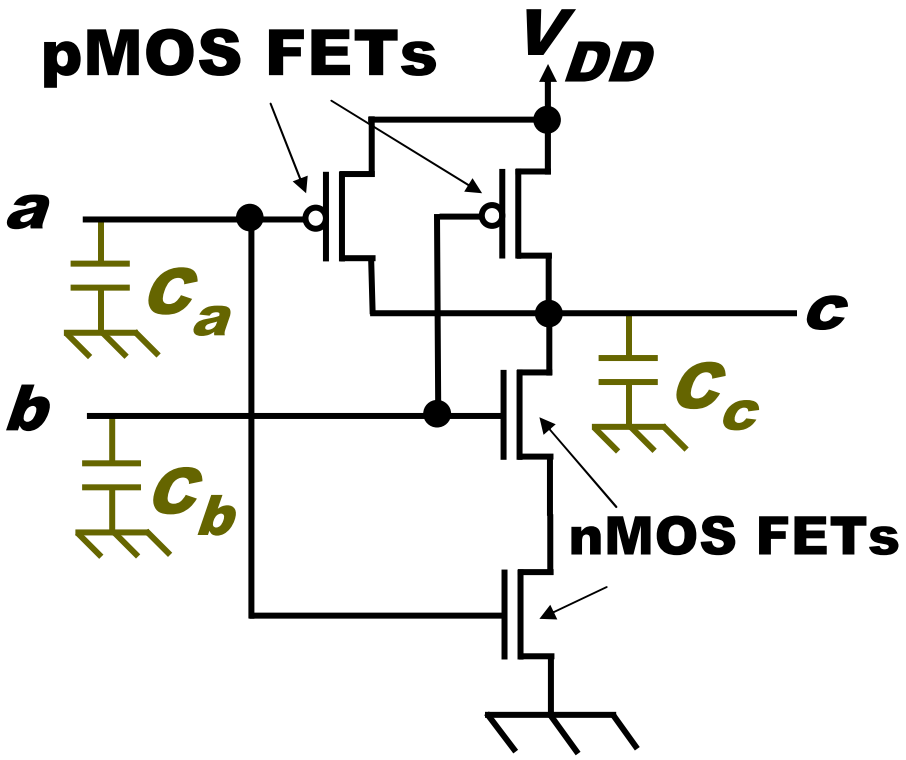
Full Adder



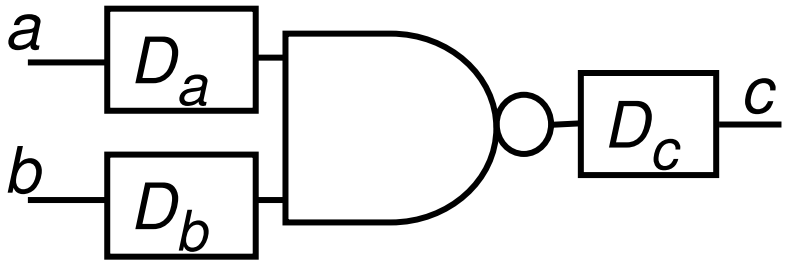
***FA**;*
***inputs**: A, B, C;*
***outputs**: Carry, Sum;*
HA: HA1, (A, B), (D, E);
HA: HA2, (E, C), (F, Sum);
OR: O2, (D, F), (Carry);

Simulation models ….Logic Model of MOS Circuit

Static CMOS NAND Gate



C_a, C_b *and* C_c are parasitic capacitances

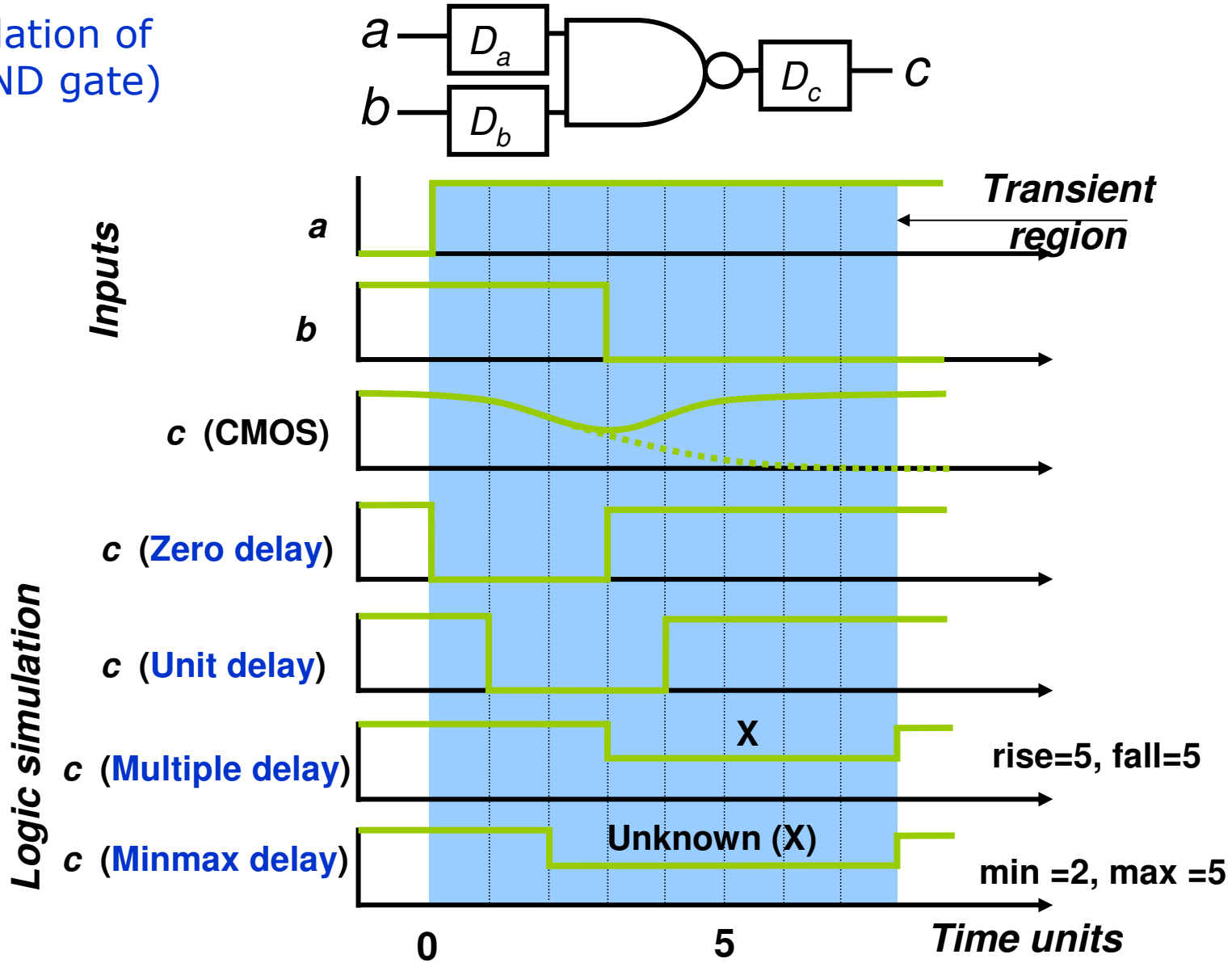


D_a and D_b are **interconnect/propagation** delays

D_c is **inertial/switching** delay of gate

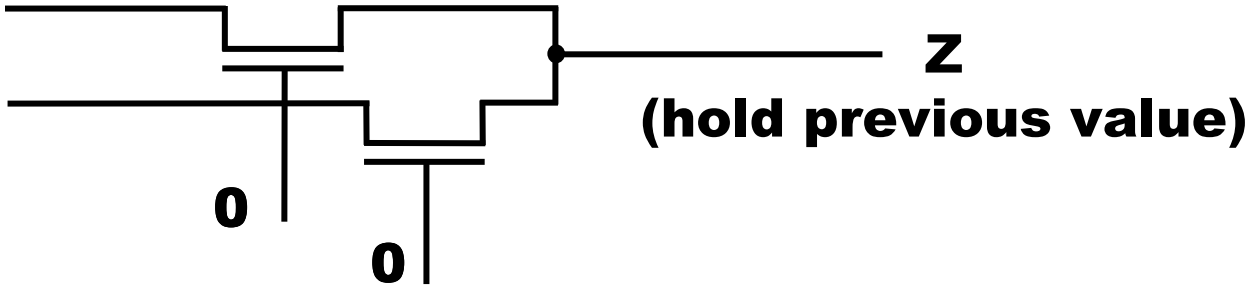
Simulation models….. Inertial/switching delay

Simulation of
a NAND gate)



Simulation models ……Modeling Signal States

- Two-states (0, 1) can be used for purely combinational logic with zero-delay.
- Three-states (0, 1, X) are essential for timing hazards and for sequential logic initialization.
  - X: unknown value
- Four-states (0, 1, X, Z) are essential for MOS devices.
  - Z: High impedance state (floating state)



- Analog signals are used for exact timing of digital logic and for analog circuits.

Logic simulation

- **True-value** simulation algorithms
  - **Compiled-code simulation**
  - **Event-driven simulation**

Logic simulation.. …Compiled code simulation

- Circuit described in a language that can be compiled & executed (e.g. HDL, C)
  - Signals are treated as variables
  - Functions like AND, OR, etc are converted to statements
  - High level functions (e.g., memory, Mux, ..) are modeled as subroutines
  - Flip-Flops are modeled as data variables

Logic simulation……Compiled code simulation

- Very effective for zero-delay combinational logic (0,1 states)
  - Cycle based simulation
- Also used for cycle-accurate synchronous sequential circuits for logic verification
  - Initialization of FFs required
- ***Efficient for highly active circuits***
- But inefficient for low-activity circuits
  - Generally 1-10% of signal activity in digital circuits
- Timing problems (e.g., glitches) are not modeled
- Allows inputs change only when circuit is stable
  - i.e., allows only synchronous circuit.

Logic simulation……Compiled code simulation

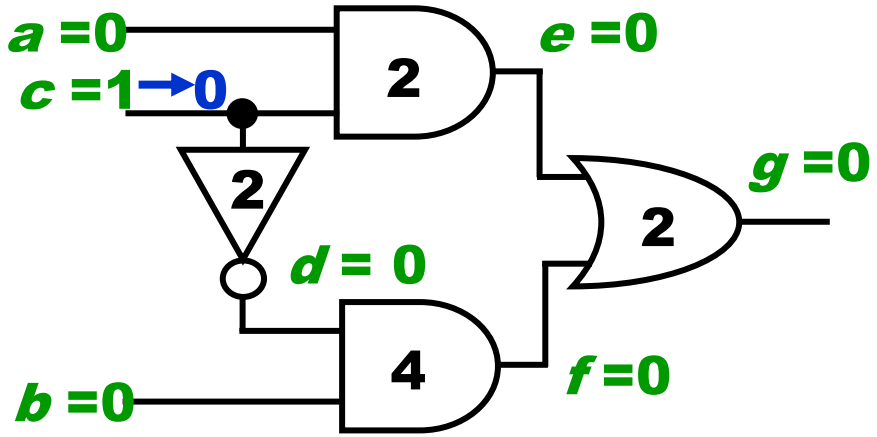
Algorithm:

- Step 1
  - Levelize combinational logic and encode in a compilable programming language
- Step 2:
  - Initialize data variables (flip-flops)
- Step 3:
  - For each input vector
    - Set primary input variables
    - Repeat (until steady-state or max. iterations)
      - Execute compiled code
      - Report or save computed variables

Logic simulation……Event-driven simulation

- Perform gates or modules evaluation only when necessary (input event)
  - *Event means a signal change*
- Simulator follows the path of events
  - Activity list
- Only the necessary amount of work is done

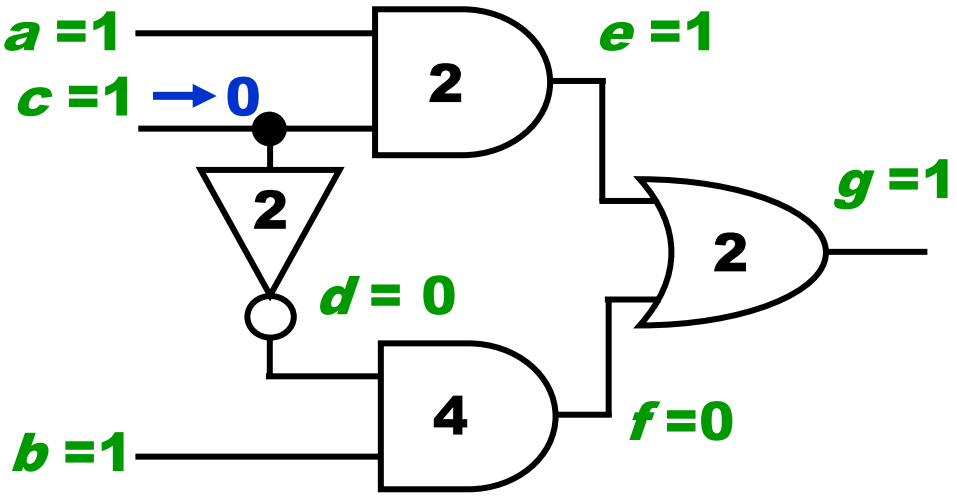
  ⇒ significant saving of computer effort



Logic simulation……Event-driven simulation

- **Efficient for low-activity circuits**
  - Lower power circuits
- Accurate implementation of general delays models
  - Timing verification
  - Event scheduling
- Best approach for circuit debug
- Best in detecting hazards
- Can be extended for fault simulation

Logic simulation……Event-driven simulation

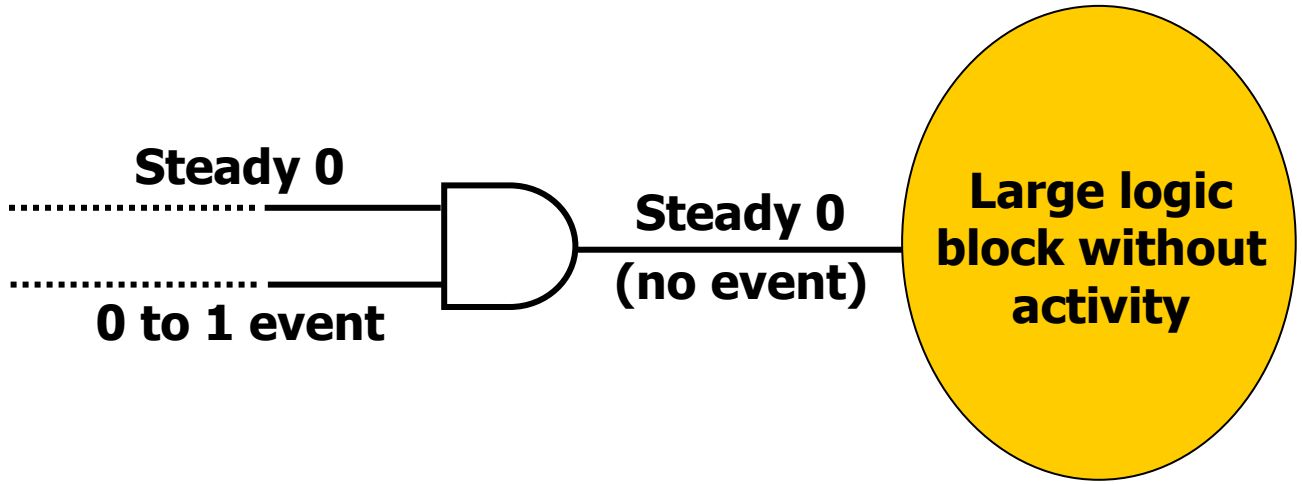


Time stack	Scheduled events	Activity list
t = 0	*c* = 0	*d*, *e*
1		
2	*d* = 1, *e* = 0	*f*, *g*
3		
4	*g* = 0	
5		
6	*f* = 1	*g*
7		
8	*g* = 1	

Logic simulation……Event-driven simulation

Efficiency

- Simulates events (value changes) only
- Speed up over compiled-code can be ten times or more; in large logic circuits about 0.1 to 10% gates become active for an input change



Summary

- Logic or true-value simulators are essential tools for design verification.
- Verification vectors and expected responses are generated (often manually) from specifications.
- A logic simulator can be implemented using either compiled-code or event-driven method.
- Per vector complexity of a logic simulator is approximately linear in circuit size.
- Modeling level determines the evaluation procedures used in the simulator.

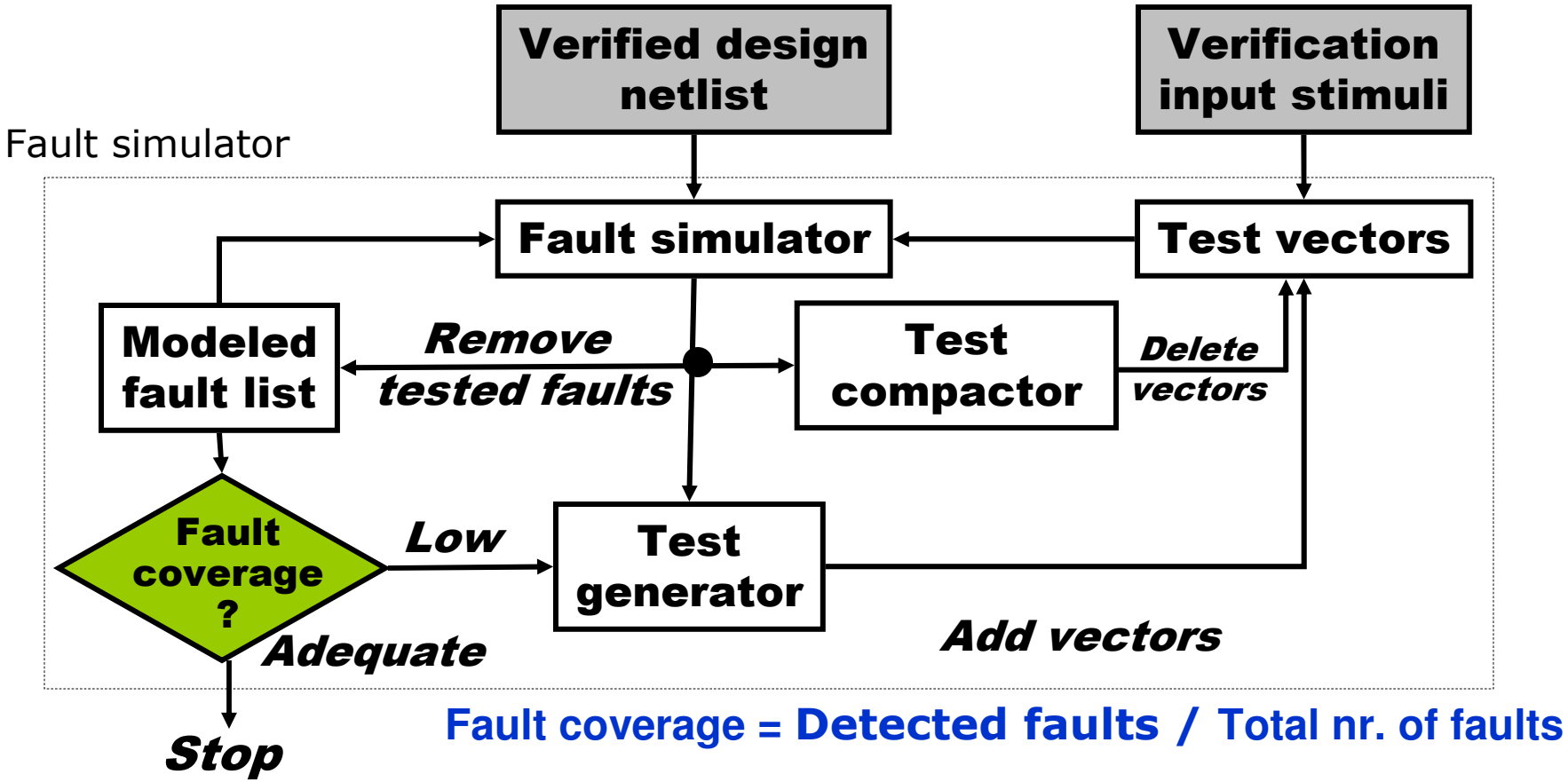
Fault Simulation

- Problem and motivation
- Simulation for Test
- Fault simulation algorithms
  - Serial
  - Parallel
  - Deductive
  - Concurrent
- Alternatives to fault simulation
  - Random Fault Sampling
- Summary

Problem and Motivation

- Fault simulation Problem
  - Given:
    - A circuit
    - A sequence of test vectors + fault model
  - Determine
    - Fault coverage
      - Fraction (or percentage) of modeled faults detected by test vectors
    - Set of undetected faults
    - (identify vectors without any added value)
    - (With help of other programs, a minimal test set for a given fault coverage for manufacturing test)
- Motivation
  - Determine test quality and in turn product quality
  - Development of manufacturing test program

Simulation for Test



- Deals with the behavior of **fabricated circuit**
- It determine the FC of each input vector
- It can generate the required test set for a given FC

Simulation for Test …….Scenario

- Circuit model: mixed-level
  - Mostly logic with some switch-level for high-impedance (Z) and bidirectional signals
  - High-level models (memory, etc.) with pin faults
- Signal states: logic
  - Two (0, 1) or three (0, 1, X) states for purely Boolean logic circuits
  - Four states (0, 1, X, Z) for sequential MOS circuits
- Timing:
  - Zero-delay for combinational and synchronous circuits
  - Mostly unit-delay for circuits with feedback

Simulation for Test …….Faults

- Mostly single stuck-at faults
- Sometimes stuck-open, transition, and path-delay faults
  - Analog circuit fault simulators are not yet in common use
- Equivalence fault collapsing of single stuck-at faults
- **Fault-dropping** -- a fault once detected is dropped from consideration as more vectors are simulated; fault-dropping may be suppressed for **diagnosis**
- **Fault sampling** -- a random sample of faults is simulated when the circuit is large

Fault Simulation Algorithms

Fault simulation time is much greater than that of design verification!

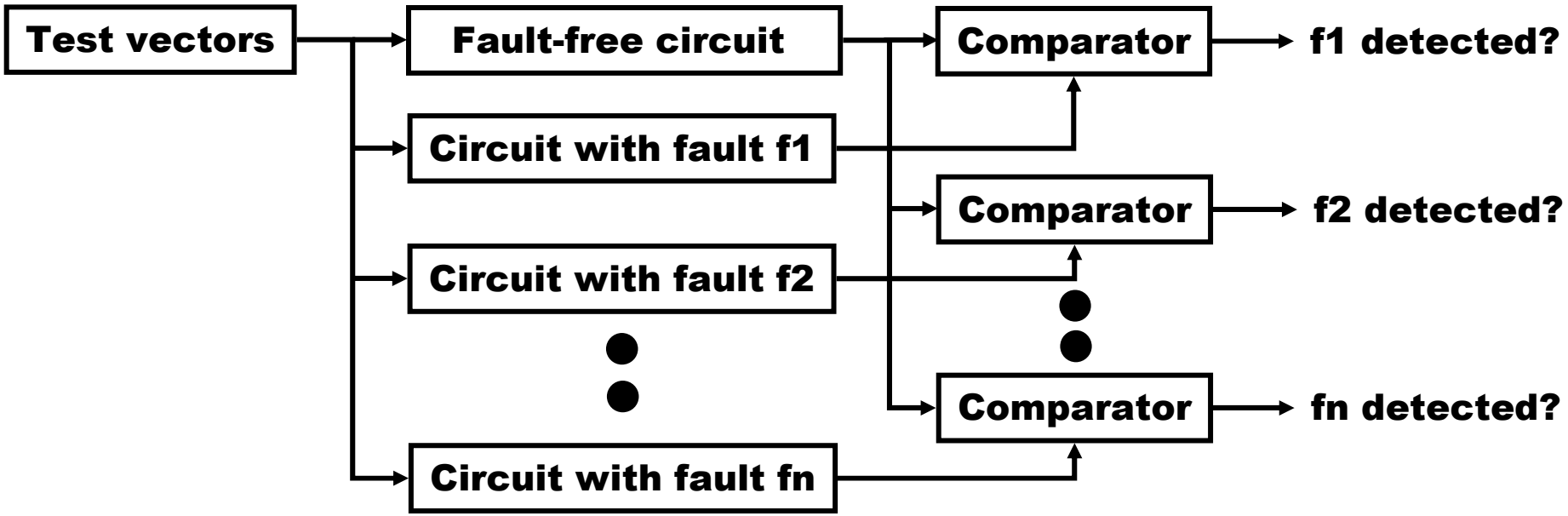
- **Serial**
  - fault free simulation + fault injection & simulation for each fault
- **Parallel**
  - Use bit-parallelism of logical operation in a digital computer
- **Deductive**
  - Deduce all signal values in each faulty circuit from simulated fault-free circuit
- **Concurrent**
  - Simulating the fault-free circuit and concurrently simulating the faulty circuit only if the faulty circuit's activity actually differs from that of the fault-free circuit (Event driven)

Fault Simulation ……Serial Algorithm

- Algorithm:
  - Simulate fault-free circuit and save responses.
  - Repeat following steps for each fault in the fault list:
    - Modify netlist by injecting one fault
    - Simulate modified netlist, vector by vector, comparing responses with saved responses
    - If response differs, report fault detection and suspend simulation of remaining vectors
- Advantages:
  - Easy to implement; needs only a true-value simulator, less memory
  - Can simulate any fault that can injected in the circuit description (e.g., Stuck-at-faults, bridging faults, stuck-open fault, delay fault, analog faults)

Fault Simulation ……Serial Algorithm

- Disadvantage:
  - Much repeated computation; CPU time prohibitive for VLSI circuits
  - For *n* faults, the CPU time may be *n* time that of true-value (golden) simulator
- Alternative: Simulate many faults together



Fault Simulation …… Parallel Algorithm

- Exploits inherent bit-parallelism of logical operations on computer words
- Multi-pass simulation: Each pass simulates *w*-1 new faults, where *w* is the machine word length
  - E.g., w=32 bit word;
  - Simulate 31 faulty circuits and a golden circuit in parallel
- Single fault injected and simulated in parallel for the same patterns
- Most effective when:
  - Signals assumed to take two-states (0,1)
  - Gates assumed to all have the same delay (zero or unit)

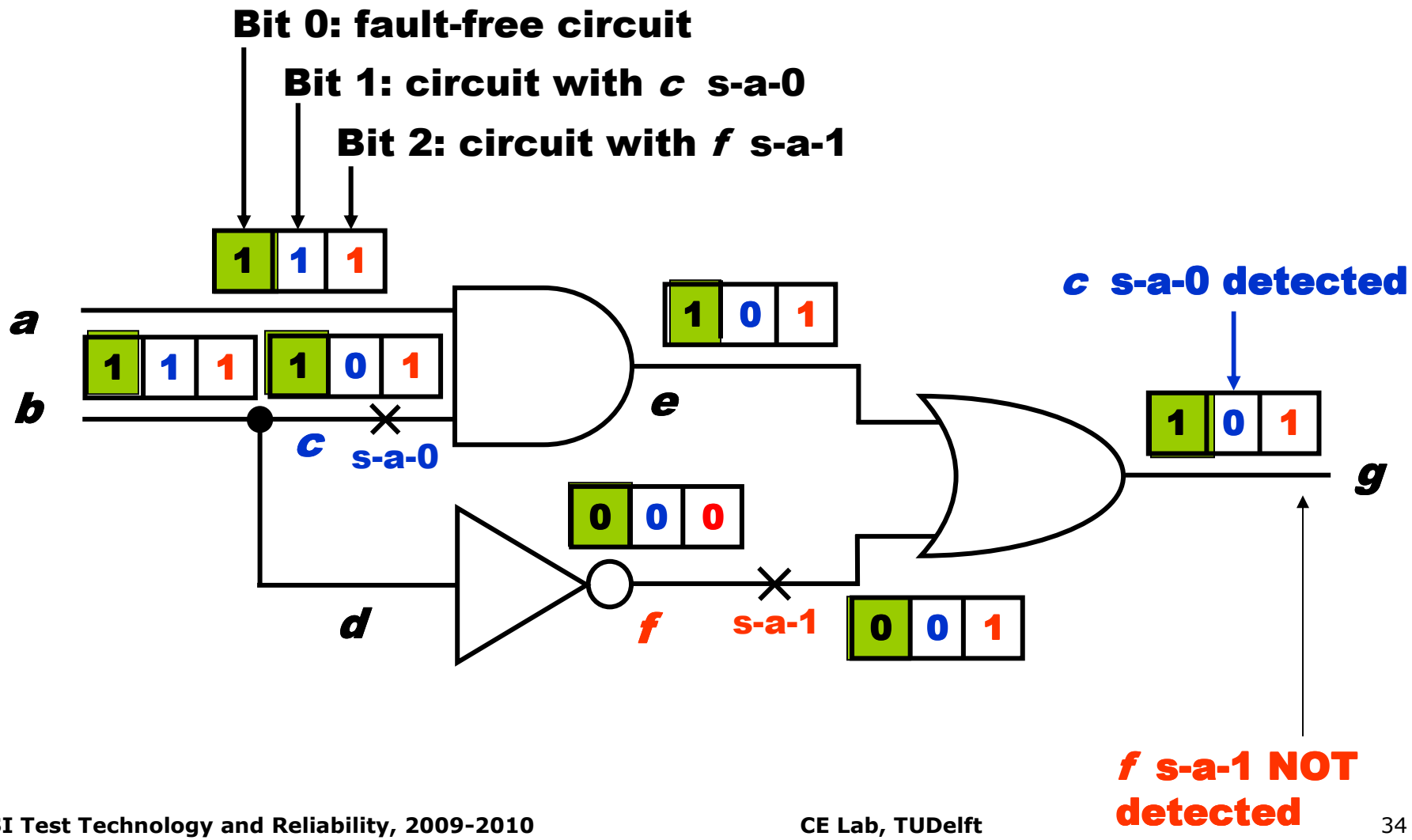
Fault Simulation …… Parallel Algorithm

- Storage: one word per line for two-state simulation
- Very fast in practice for combinational circuits.
  - Speed up over serial method ~ *w*-1
- Not suitable for circuits with timing-critical and non-Boolean logic
- Compiled code or even driven simulation

Fault Simulation …… Parallel Algorithm

Example:

Assume a computer with three bit word



Fault Simulation …… Deductive Algorithm

- One-pass simulation: Only fault free circuit simulated
  - Explicitly simulating only the behavior of the fault-free circuit
  - Simultaneously deducing from the current good state of the circuit all faults that are detectable at any internal or output line
  - Following true-value simulation of each vector, fault lists of all gate output lines are **updated** using set-theoretic rules, signal values, and gate input fault lists
  - Results for each faulty circuit are deduced
- PO fault lists provide detection data
- Each line k contains a list L_k of faults detectable on k
- Circuit model:
  - Signals assumed to take two-states (0,1)
  - Boolean gates assumed to all have the same delay (or unit)

Fault Simulation …… Deductive Algorithm

- Advantages
  - Efficient in processing all faults at the same time
  - Compiled code or even driven simulation
  - Tremendous speed up

- Limitations:
  - Set-theoretic rules difficult to derive for non-Boolean gates
  - Only suitable for zero or unit delay
  - Gate delays are difficult to use
  - Unknown faults are not easy handled
  - Requiring large storage.
    - Size of fault lists cannot be predicted in advance

Fault Simulation …… Deductive Algorithm

- Rules for fault list propagation

Gate type	Inputs a	Inputs b	Output c	Output fault list L_c
AND	0	0	0	$[L_a \cap L_b] \cup c_1$
	0	1	0	$[L_a \cap \overline{L_b}] \cup c_1$
	1	0	0	$[\overline{L_a} \cap L_b] \cup c_1$
	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	$[\overline{L_a} \cap L_b] \cup c_0$
	1	0	1	$[L_a \cap \overline{L_b}] \cup c_0$
	1	1	1	$[L_a \cap L_b] \cup c_0$
NOT	0	-	1	$L_a \cup c_0$
	1	-	0	$L_a \cup c_1$

Fault Simulation …… Deductive Algorithm

Example:

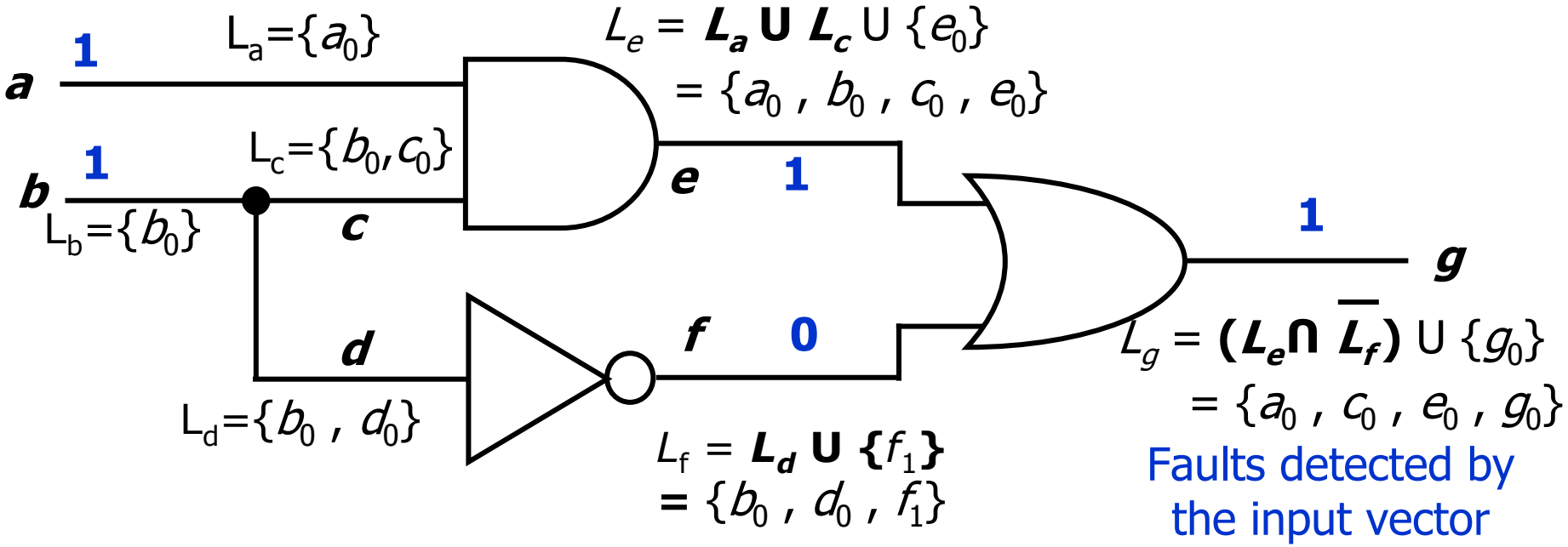
Notation: L_k is fault list for line k; k_n is s-a-n fault on line k

Applied vector is ab=11

Simulate all s-a-1 and s-a-0 for all lines

First: golden simulation to determine all signal values

Second: fault list generation and propagation



Fault Simulation …… Concurrent Algorithm

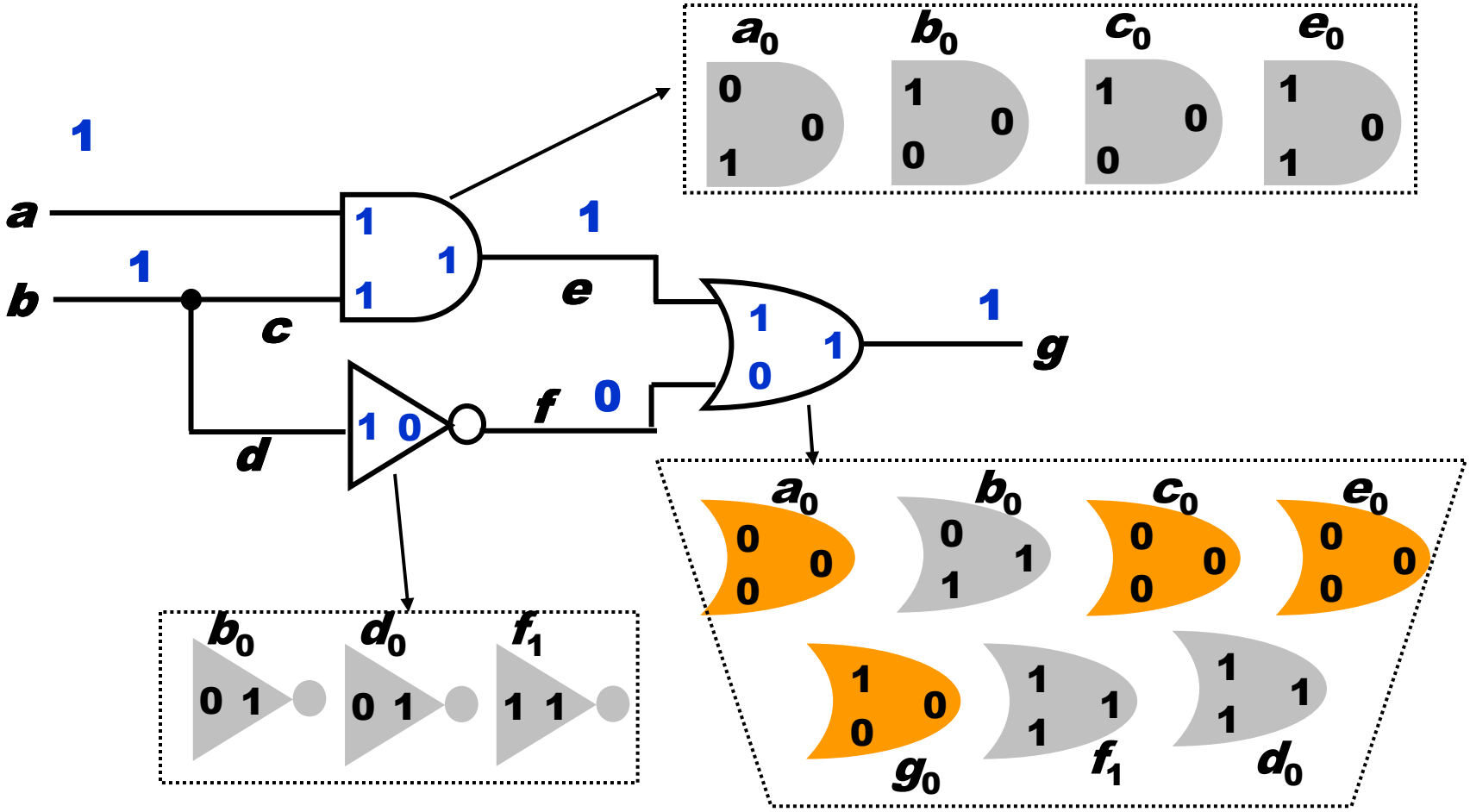
- **Event-driven simulation** of fault-free circuit and (concurrently) only those parts of the faulty circuit that differ in signal states from the fault-free circuit.
  - Event scheduling and processing
- All events of fault-free and all faulty circuits are implicitly simulated.
- A list per gate containing copies of the gate from all faulty circuits in which this gate differs. List element contains fault ID, gate input and output values and internal states, if any.

Fault Simulation …… Concurrent Algorithm

- Most general method of fault simulation
- Handles various types of circuits models, faults, signal states and timing models (offers most flexibility.)
- Directly applicable to functional-level and mixed-level modeling.
- Requiring even more memory space.

Fault Simulation …… Concurrent Algorithm

- Perform the golden simulation with 11 as input
- To each good gate, a number of bad gates are attached
- At the PO, any bad gate whose output is different from of good gate indicates a fault detection



Alternatives to fault simulation

Fault Sampling

- Popular technique to reduce fault simulation effort
- A randomly selected subset (***sample***) of faults is picked and simulated.
- A **sample coverage** is measured is used to estimate fault coverage in the entire circuit.
  - Accuracy depends on the **sample size**.
- Advantage:
  - Saving in computing resources (CPU time and memory.)
- Disadvantage:
  - Limited data on undetected faults.

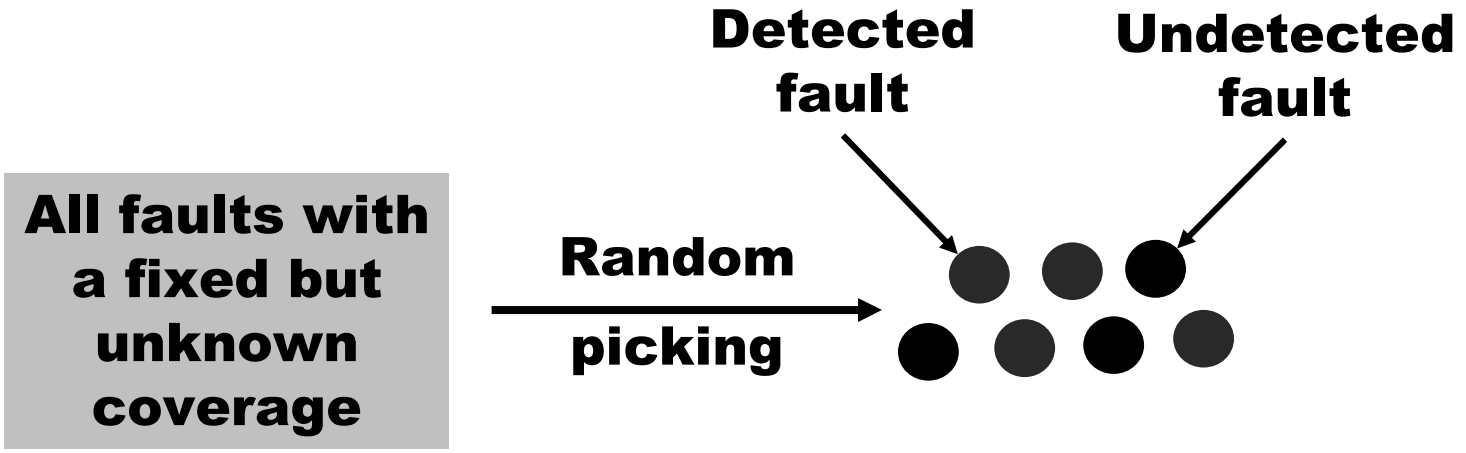
Alternatives to fault simulation

Motivation for Sampling

- Complexity of fault simulation depends on:
  - Number of gates
  - Number of faults
  - Number of vectors
- Complexity of fault simulation with fault sampling depends on:
  - Number of gates
  - Number of vectors
- The method has significant advantages in reducing CPU time and memory needs of the simulator.

Alternatives to fault simulation

- Random Sampling Model



N_p = total number of faults (population size)

C = fault coverage (unknown)

N_s = sample size

$N_s << N_p$

c = sample coverage (a random variable)

Summary

- Fault simulator is an essential tool for test development.
- Four fault simulation algorithms
  - Serial, Parallel, Deductive, Concurrent
- Concurrent fault simulation algorithm offers the best choice.
- For large circuits, the accuracy of random fault sampling only depends on the sample size (1,000 to 2,000 faults) and not on the circuit size. The method has significant advantages in reducing CPU time and memory needs of the simulator.

- Next lecture – we will talk about Testability measures and Automated Test Pattern Generation (ATPG) for Combinational Circuits