ECE 538

VLSI System Testing

Krish Chakrabarty

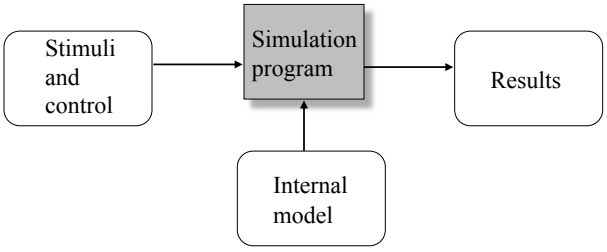
Logic Simulation

Introduction

- Motivation
- Types of logic simulation
  - Compiled code
  - Event-driven
- Delay models
- Element evaluation
- Hazard detection

Motivation

- A design verification technique (functional and timing)
- Compare results obtained with expected responses specified by the specification
- Use software model



Motivation

- Correctness, independent of initial (power-on) state
- Insensitive to small variations in component delays
- Free of races, oscillations, "illegal" input combinations, "unsafe" states
- Evaluation of design alternatives (what-if scenarios)
- Documentation (generation of timing diagrams)

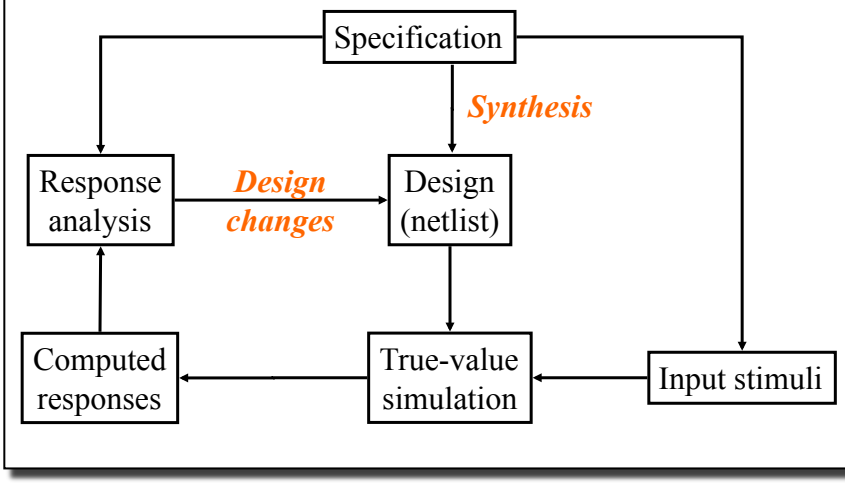
Logic Simulation

- What is simulation?
- Design verification
- Circuit modeling
- True-value simulation algorithms
  - Compiled-code simulation
  - Event-driven simulation
- Summary

Simulation Defined

- Definition: Simulation refers to modeling of a design, its function and performance.
- A software simulator is a computer program; an emulator is a hardware simulator.
- Simulation is used for design verification:
  - Validate assumptions
  - Verify logic
  - Verify performance (timing)
- Types of simulation:
  - Logic or switch level
  - Timing
  - Circuit
  - Fault

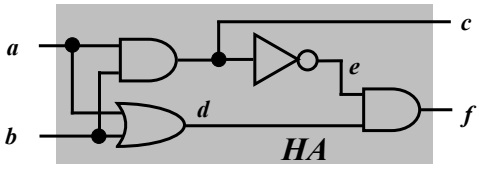
Simulation for Verification



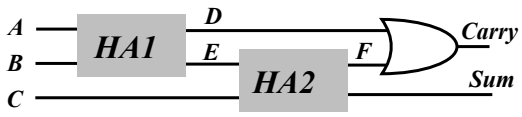
Modeling for Simulation

- **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.**

Example: A Full-Adder

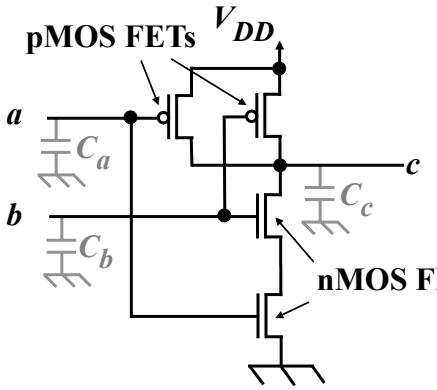



```
HA;
inputs: a, b;
outputs: c, f;
AND: A1, (a, b), (c);
AND: A2, (d, e), (f);
OR: O1, (a, b), (d);
NOT: N1, (c), (e);
```

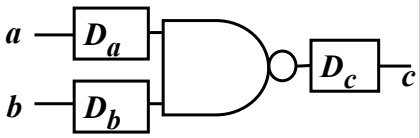



```
FA;
inputs: A, B, C;
outputs: Carry, Sum;
HA: HA1, (A, B), (D, E);
HA: HA2, (E, C), (F, Sum);
OR: O2, (D, F), (Carry);
```


Logic Model of MOS Circuit



C_a, C_b and C_c are parasitic capacitances

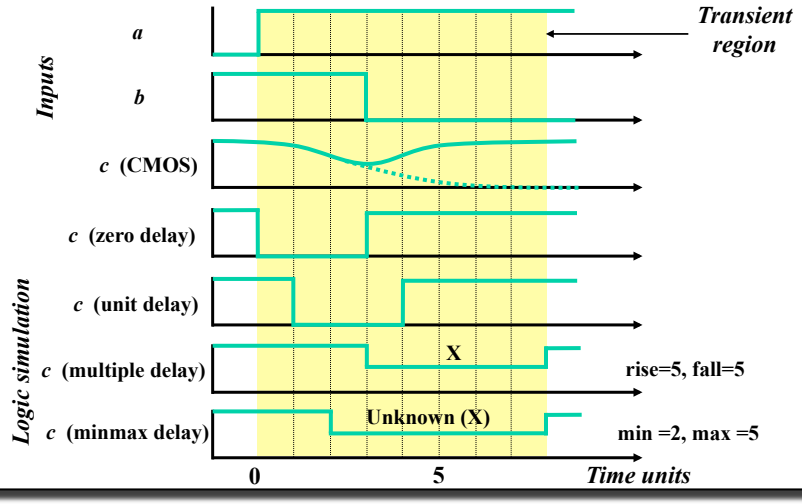


D_a and D_b are interconnect or propagation delays

D_c is inertial delay of gate

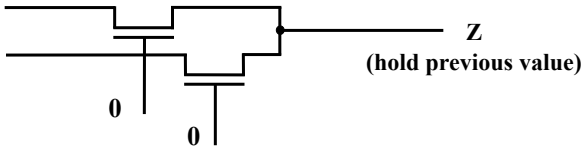
Options for Inertial Delay

(simulation of a NAND gate)



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.
- Four-states (0, 1, X, Z) are essential for MOS devices. See example below.
- Analog signals are used for exact timing of digital logic and for analog circuits.



Modeling Levels

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

True-Value Simulation Algorithms

- Compiled-code simulation (oblivious simulation)
  - Applicable to zero-delay combinational logic
  - Also used for cycle-accurate synchronous sequential circuits for logic verification
  - Efficient for highly active circuits, but inefficient for low-activity circuits
  - High-level (e.g., C language) models can be used
- Event-driven simulation (exclusive simulation of activity)
  - Only gates or modules with input events are evaluated (*event means a signal change*)
  - Delays can be accurately simulated for timing verification
  - Efficient for low-activity circuits
  - Can be extended for fault simulation

Types of Simulation (Contd.)

- Compiled-code (oblivious)
  - The circuit is described in a programming language and an executable model is generated
  - Circuit operation ≡ program execution
  - Fast and efficient but inflexible; practical only for small circuits
- Event-driven
  - Exclusive simulation of activity
  - Circuit is a data structure, simulation program is same for all circuits
  - Flexible, but requires event list management (overhead)

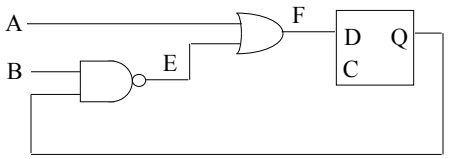
Compiled-Code Algorithm

- Step 1: Levelize combinational logic and encode in a compilable programming language
- Step 2: Initialize internal state 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

Compiled-Code Simulation

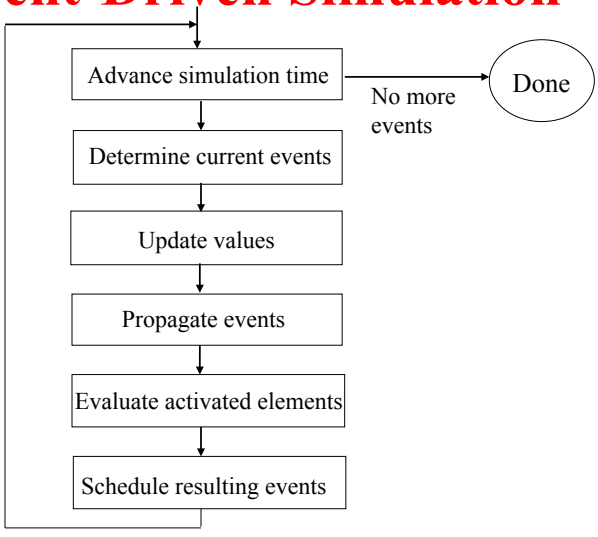
```
LDA  B
AND  Q
INV
STA  E
OR   A
STA  F
STA  Q
```

Simulation program

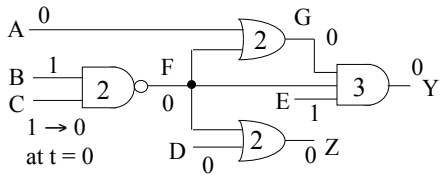


- Delays can be modeled by explicitly adding them to the software model

Event-Driven Simulation



Event-Driven Simulation



Snapshot of event list

t = 0

Event	Time
F = 1	t = 2

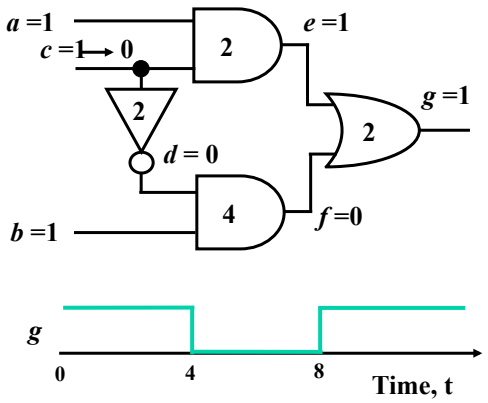
t = 2

Event	Time
G = 1	t = 4
Z = 1	t = 4

t = 4

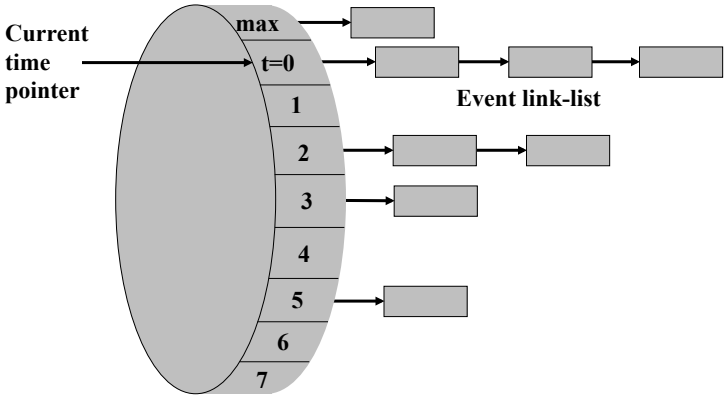
Event	Time
Y = 1	t = 7

Event-Driven Algorithm (Example)



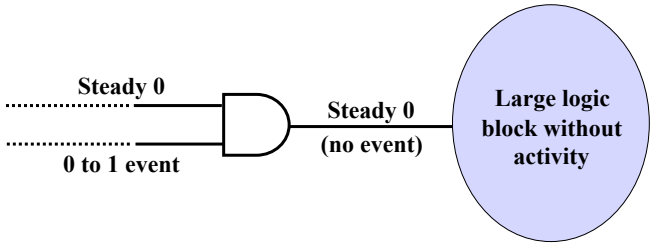
Time stack	Scheduled events	Activity list
t = 0	*c* = 0	*d, e*
1		
2	*d* = 1, *e* = 0	*f, g*
3		
4	*g* = 0	
5		
6	*f* = 1	*g*
7		
8	*g* = 1	

Time Wheel (Circular Stack)



Efficiency of Event-driven Simulator

- 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

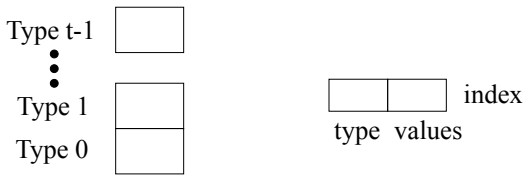


Delay Models

- Transport delay (pure delay)
  - Single delay value for each gate
- Rise/fall times
  - Two delay values for each gate output
- Ambiguity delay
  - Min and max delay values for each gate
  - Ambiguity region becomes enlarged towards primary outputs
- Inertial delay
  - Filter out narrow pulses (spikes and glitches)

Element Evaluation

- Simulation speed depends on fast elements (gates) that can be evaluated
- Truth tables
  - O(1) evaluation but $O(2^N)$ storage
  - Decision step needed (which truth table?)
- Zoom table of size tS combines t individual truth tables, S is size of largest truth table



Input Scanning

```
evaluate (G,c,i)
begin
  u_values = FALSE;
  for every input value of G
   begin
     if v = c then return c ⊕ i
     if v = u then u_values = TRUE
   end
  if u_values return u
  return c̄ ⊕ i
end
```

O(1) storage, O(N) evaluation

- Controlling value c
- Inversion value i

c	X	X	c ⊕ i
X	c	X	c ⊕ i
X	X	c	c ⊕ i
c̄	c̄	c̄	c̄ ⊕ i

	c	i
AND	0	0
OR	1	0
NAND	0	1
NOR	1	1

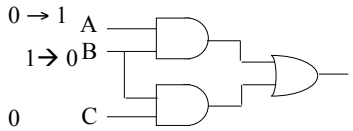
Input Counting

- Maintain two counters *c_count* and *u_count* for every gate
  - *c_count*: number of inputs with c values
  - *u_count*: number of inputs with u values
- These counters must be updated

```
evaluate (G,c,i)
begin
  if c_count > 0 then return c ⊕ i
  if u_count > 0 then return u
  return c̄ ⊕ i
end
```

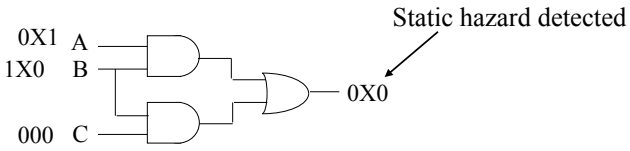
- O(1) storage
- O(1) evaluation

Hazard Detection



Static hazard: 0 → 1 → 0
Dynamic hazard:
0 → 1 → 0 → 1

- Simulator should detect hazards (analyze transient behavior of signals)
- 3-valued logic simulation:



- How many logic values are necessary?
- More the better, but…more complexity

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.