ECE-470 Digital Design II
Logic Simulation

Outline

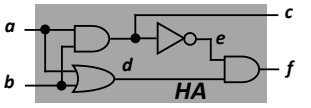
- **Logic Modeling**
  - Model types
  - Models at different levels of abstractions
  - Logic models and definitions
- **Logic Simulation**
  - What is simulation?
  - Design verification
  - Circuit modeling
  - Determining signal values
  - True-value simulation algorithms
  - Compiled-code simulation
  - Event-driven simulation

Logic Modeling: Model Types

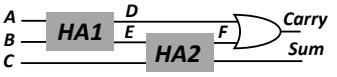
- Functional
  - DC behavior – no timing
- Behavioral
  - System at I/O level
  - Timing information is provided
  - Internal details missing
- Structural
  - Gate level description
  - External representation (used by user)
  - Internal representation (inside a computer)

Models are often described using an **hierarchy**

Hierarchical Model: 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);*

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

Logic Models and Definitions

- Combinational circuit models
  - Function expressed as truth-table or cubes
  - Cubes and cube intersection can be used during simulation
- Sequential Circuits
  - Structure represented as a collection of flip-flops feeding combinational logic
  - Time frame expansion is possible
- Binary Decision Diagrams (BDD)

Logic Models and Definitions

- Program model of a circuit
  - Express circuit (gate level) as a program consisting of interconnected **logic operations**
  - Execute the program to determine circuit output for varying inputs
- RTL model
  - Higher level model of the circuit
- HDL model
  - Examples at this level: Verilog, VHDL

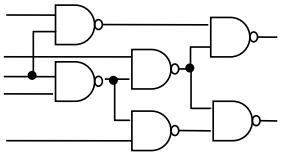
Logic Models and Definitions

- Structural model
  - **External representation** in the form of **netlist**
  - Examples of this are: UW, ISCAS, BLIF, etc.
  - Keywords used in such representations:
    - Primary Inputs (PI) and Primary Outputs (PO)
    - Gates: AND, OR, NOT, etc.
    - Storage: latch, flip-flop
    - Connections: lines, nets
    - Fanin: number of inputs to a gate
    - Fanout : number of lines a signal line feeds
    - Fanout free circuit: every line or gate has a fanout of one

Example: BLIF netlist format

BLIF format, mapped using SIS tool with a library of gates

```
.model C17.iscas
.inputs 1GAT(0) 2GAT(1) 3GAT(2) 6GAT(3) 7GAT(4)
.outputs 22GAT(10) 23GAT(9)
.default_input_arrival 0.00 0.00
.default_output_required 0.00 0.00
.default_input_drive 1.98 1.82
.default_output_load 0.10
.gate nand2 a=1GAT(0) b=3GAT(2) O=10GAT(6)
.gate nand2 a=3GAT(2) b=6GAT(3) O=11GAT(5)
.gate nand2 a=11GAT(5) b=2GAT(1) O=16GAT(8)
.gate nand2 a=10GAT(6) b=16GAT(8) O=22GAT(10)
.gate nand2 a=11GAT(5) b=7GAT(4) O=19GAT(7)
.gate nand2 a=16GAT(8) b=19GAT(7) O=23GAT(9)
.end
```



Logic Models and Definitions

- Structural model
  - **Internal representation** in the form of **tables**
    - Tables of gates and storage elements (names)
    - Tables of connections
    - Tables of fanin and fanouts
  - Objective is to make the storage and search processes (integral part of simulation) more efficient
  - Knowledge of data structures and algorithms is very useful

Logic Models and Definitions

- Additional useful terms
  - Graph representation
  - Reconvergent fanouts
  - Stems and branches
  - Logic level/depth in a circuit
  - "levelization" of a circuit

Logic Simulation

Motivation

- **Logic simulation** is used to verify the correctness of the design and tests
- It avoids building costly hardware
- Can help debug a design in many more ways than the real hardware could
- Understanding simulation will help understand the limitations of the simulation process and the simulator in question

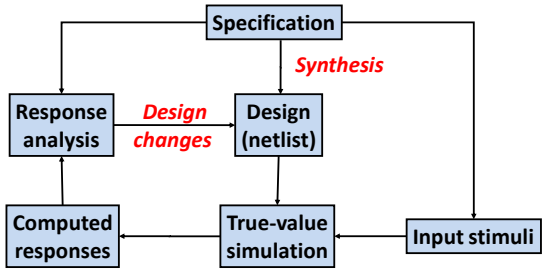
13

Simulation Defined

- **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)
- **Simulation is used for test generation**
- Types of simulation:
  - Logic or switch level
  - Timing
  - Circuit
  - Fault

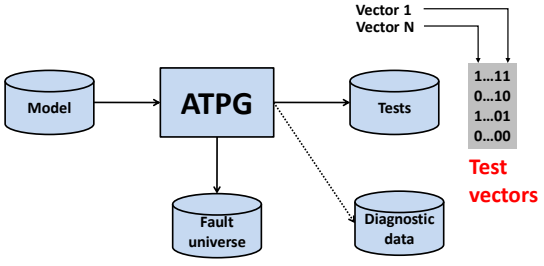
14

Simulation for Verification



15

Simulation for Test Generation



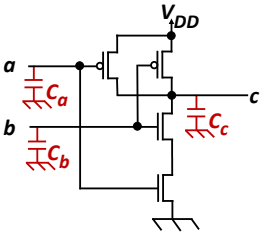
16

Modeling for Simulation

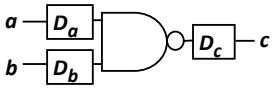
- Modules, blocks or components described by
  - Input/output (I/O) function
  - Delays associated with I/O signals
  - Examples: binary adder, Boolean gates, resistors and capacitors
- Interconnects represent
  - Ideal signal carriers, or
  - Ideal electrical conductors
- **Netlist**: a format (or language)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

17

Logic Model of MOS Circuit



C_a, C_b and C_c are parasitic capacitances



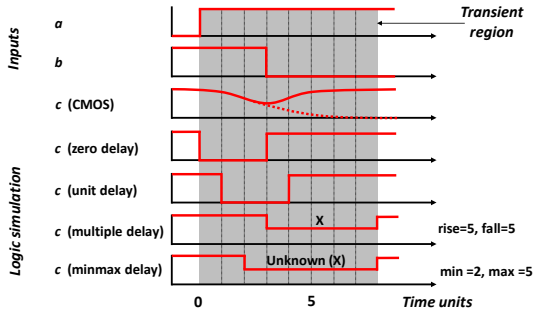
D_a and D_b are interconnect or propagation delays

D_c is inertial delay of gate

18

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
- Analog signals are used for exact timing of digital logic and for analog circuits
- Determining gate values:
  - Use of software logic primitives such as AND, OR, NOT instructions
  - Search the truth table
  - Use cubes and cube intersection rules for processing

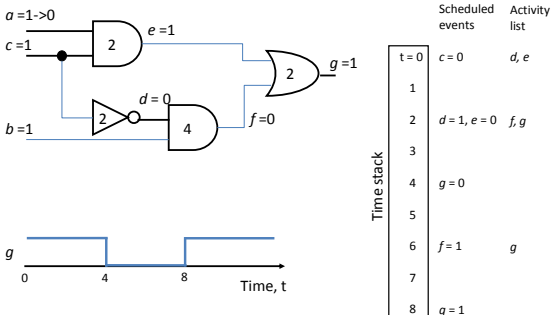
True-value Simulation Algorithms

- **Compiled-code simulation**
  - Applicable to zero or constant 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**
  - 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**

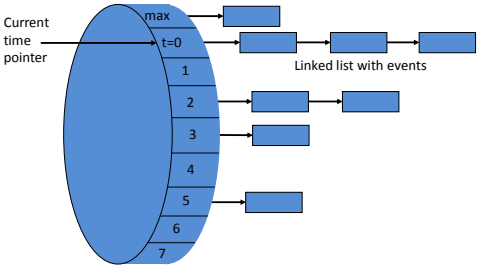
Compiled-code Algorithm

- **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

Event-driven Algorithm

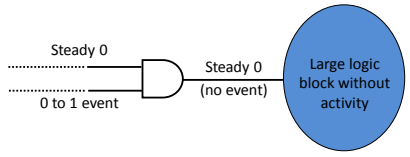


Time Wheel



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



25

Summary

- Logic or true-value simulators are essential tools for design verification
- 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

26