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

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

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.lib
.inputs 1GAT(0) 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=16GAT(8) b=16GAT(8) O=23GAT(9)
.gate nand2 a=11GAT(5) b=7GAT(4) O=16GAT(10)
.gate nand2 a=16GAT(10) 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
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

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

Simulation for Verification

- Specification
- Design (netlist)
- Computed responses
- True-value simulation
- Input stimuli

Simulation for Test Generation

- Model
- ATPG
- Fault universe
- Diagnostic data
- Test vectors

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

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)

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Logic simulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td></td>
</tr>
<tr>
<td>c (CMOS)</td>
<td></td>
</tr>
<tr>
<td>c (zero delay)</td>
<td></td>
</tr>
<tr>
<td>c (unit delay)</td>
<td></td>
</tr>
<tr>
<td>c (multiple delay)</td>
<td></td>
</tr>
<tr>
<td>c (minmax delay)</td>
<td></td>
</tr>
</tbody>
</table>

**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**

- Current time pointer
- Linked list with events

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

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