ECE 553: TESTING AND TESTABLE DESIGN OF DIGITAL SYSTEMS

Overview

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

Motivation

- Logic simulation is used to verify or ascertain assertions (design, device, …)
- 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(s) in question

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

- Specification
- Design (netlist)
- Design changes
- Computed responses
- True-value simulation
- Input stimuli
- Response analysis
- Synthesis

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.
**Logic Model of MOS Circuit**

- pMOS FETs
- nMOS FETs
- $C_a$, $C_b$, 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**

<table>
<thead>
<tr>
<th>Simulation</th>
<th>Time Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>(simulation of a NAND gate)</td>
<td></td>
</tr>
</tbody>
</table>

- Transient region
- $a$, $b$, and $c$ (CMOS)
- $c$ (zero delay)
- $c$ (unit delay)
- $c$ (multiple delay)
- $c$ (minmax delay)

**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.

**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
- Use of X value and its processing
  - Example: x value simulation, problems associated with it, possible fixes and conservative processing

**True-Value Simulation Algorithms**

- Compiled-code 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
  - 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 (Example)

<table>
<thead>
<tr>
<th>Scheduled events</th>
<th>Activity list</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a = 1$</td>
<td>$c = 0$</td>
</tr>
<tr>
<td>$c = 1$</td>
<td>$d, e$</td>
</tr>
<tr>
<td>$d = 0$</td>
<td>$f = 0$</td>
</tr>
<tr>
<td>$b = 1$</td>
<td>$g = 0$</td>
</tr>
<tr>
<td>$g = 1$</td>
<td>$t = 0$</td>
</tr>
</tbody>
</table>

$\text{Time stack}$

0 1 2 3 4 5 6 7 8

$\text{Current time pointer}$

$\text{Event link list}$

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

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.