Design Verification
Lecture 01

- Course Title: Verification of Digital Systems
- Professor: Michael Hsiao (355 Durham)
- Prerequisites: Digital Logic Design, C/C++ Programming, Data Structures, Computer Organization/Architecture, Discrete Math
- Course website: computing.ece.vt.edu/~mhsiao/ece5506

1. Course Outline

- Overview + background in discrete math, graphs, algorithms, complexity
- Combinational circuit verification
  - 2-level verification based on containment
  - Multi-level verification using SAT
  - Multi-level verification using BDDs
  - Incremental verification
- Sequential circuit verification
  - Projection and reachability computation
  - Image/pre-image computation
  - Approximate techniques
  - Incremental verification
- Model checking
  - Computational Tree Logic (CTL)
  - Property checking
  - Symbolic and bounded model checking
- Simulation-based verification
  - Coverage metrics
  - Error-directed verification
  - Symbolic simulation
- Architectural verification
- Pipeline checking
- Design for verifiability

- Error Diagnosis
  - Static diagnosis
  - Dynamic diagnosis

2. Grading

- Homework: 
- Exam 1: 
- Exam 2: 
- Project: 

3. Activities of Verification and CAD

- Design optimization
- Reduce design time
- Large scale design management
- *Strategic importance

4. Trends in microelectronic design

- Smaller circuits (area)
- Higher performance (speed)
- More devices on a chip (complexity)
- Lower power/energy consumption
- Lower cost
- Higher reliability

5. EDA (Electronic Design Automation) Tool Issues/Problems

- Need to be competitive in performance
- Need to be competitive in price
- Time-to-market is critical
- Volume
- CAD lags behind fabrication technology
6. Types of Microelectronic Circuits

- Microprocessors (general purpose + DSP)
  - high-volume (e.g. Pentium, X86)
  - high-performance (e.g. HP C-180, Sun Ultra-II, Titanium, etc.)
- Application Specific IC’s (ASIC’s)
  - Varying volume and performance
- Embedded systems
- System-on-a-chip
- Special applications (e.g. NASA)
- Prototypes (experimentation, gate-arrays, FPGA, etc.)

7. Design Styles

- Full-custom: design everything from scratch
- Semi-custom: design from a library of cells
- Pre-diffused: (MPGA) Mask Programmable Gate-Array
- Pre-wired: (FPGA) Field Programmable Gate-Array

8. Design Style Trade-offs

<table>
<thead>
<tr>
<th></th>
<th>Full Custom</th>
<th>Semi-Custom</th>
<th>MPG A</th>
<th>FPGA</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Density</strong></td>
<td>Very high</td>
<td>High</td>
<td>High</td>
<td>Low to Medium</td>
</tr>
<tr>
<td><strong>Performance</strong></td>
<td>Very high</td>
<td>High</td>
<td>High</td>
<td>Low to Medium</td>
</tr>
<tr>
<td><strong>Flexibility</strong></td>
<td>Very high</td>
<td>High</td>
<td>Medium</td>
<td>Low</td>
</tr>
<tr>
<td><strong>Design Time</strong></td>
<td>Very long</td>
<td>Short</td>
<td>Short</td>
<td>Very short</td>
</tr>
<tr>
<td><strong>Cost (low-vol)</strong></td>
<td>Low</td>
<td>High</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td><strong>Cost (high-vol)</strong></td>
<td>Medium</td>
<td>Medium</td>
<td>Low</td>
<td>Medium to high</td>
</tr>
<tr>
<td><strong>Manufact. time</strong></td>
<td>Medium</td>
<td>Medium</td>
<td>Short</td>
<td>Very short</td>
</tr>
</tbody>
</table>

9. Circuit Production Flow

(a) Design: modeling → (synthesis + optimization ↔ verification + validation)
(b) Testing: Stimuli generation
(c) Fabrication: Mask generation + wafer generation
(d) Packaging

We will focus on verification+validation for this course
10. Typical Design+Verification Flow

11. Examples of Verification Objectives

- Verify correctness of specification
- Verify correctness of protocol (communications, coherence, etc.)
- Verify a set of properties the design should hold
- Verify equivalence between RTL and gate-level
- Verify equivalence between gate-level and optimized gate-level
- Verify timing/performance of the design against spec
- Verify power consumption of the design (avg power, peak power) against spec
- Verify equivalence between gate-level and gate-level extracted from (full-custom) transistor/layout

12. Modeling Views

- Behavioral view: set of tasks
- Structural view: interconnection of parts
- Physical view: physical objects with size and positions
13. Verification at different levels

- Functional Level: determine if macroscopic structure is correct
- Logic Level: determine if microscopic structure is correct
- Geometric level: determine correctness of position, dimension, and connections
- Implementation Verification: verify correctness of implementation against golden model
- Design Verification: verify correctness of conceptualization & modeling of (golden) model

14. Simulation

- Functional-level
  - simulating HDL (high-level description language) models
    * cycle-based simulators
    * transaction-based simulators
    * code-coverage simulators
  - may be based standard programming languages
  - can be 5 to 100 times faster than logic-level simulators
- Logic-level
  - logic simulation: event-driven based
  - timing simulation: event-driven based
- Circuit-level
  - SPICE, deriving voltage levels, etc.

15. Formal Verification vs. Simulation-based Verification

**Example:** verify \((x + 1)^2 = x^2 + 2x + 1\)

<table>
<thead>
<tr>
<th>(x)</th>
<th>((x + 1)^2)</th>
<th>(x^2 + 2x + 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>9</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>67</td>
<td>4624</td>
<td>4624</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
key: need to simulate enough patterns to ensure confidence of correctness

By formal proof:

Using mathematical definitions, distributivity, associativity, substitution, etc., we can prove \((x + 1)^2 = x^2 + 2x + 1\)

16. Formal Specifications

A formal specification (requirements specification) is a concise description of the behavior and properties of a system written in a mathematically-based language, stating what a system is supposed to do in the context it is supposed to operate as abstractly as possible.

Example: The following specification is given: The input accepts a stream of bits and the output emits the same stream delayed by four cycles.

- vague: does the last 4 bits include the current bit?
- incomplete: what is the output on the first, second, and third cycles?
- unusable: hard to compile into executable code for simulation.
- intractable: the text is verbose, unstructured and hard to analyse.

Better specification: \(\text{output}(t) = \text{if } (t < 4) \text{ then } F \text{ else } \text{input}(t-4)\)

17. Verification of large designs, such as SOCs or chips with many modules

- Divide and conquer (start with the smallest block)
- Option 1: formal check
  - verify low-level modules formally
  - verify interconnection of modules by checking the consistency of the interaction/communication between modules (property checking)
- Option 2: simulation-based
  - verify low-level module either by simulation or formal verification
  - verify interconnection via simulation, and check consistency

18. Lecture Notes: Please remember to print and bring lecture notes before class in the future.