046880: Logic Design Automation of VLSI Circuits

Course abstract:
The course describes issues of VLSI digital system design. As system complexity grows, designers are faced with problems related to validating correctness, reducing design effort, and optimizing speed, area, power and other parameters. The course presents solutions and principles in terms of design methodologies, design automation algorithms and tool capabilities. The generic design flow is covered top-down. The course provides a broad survey of the electronic design automation (EDA) field for a complete perspective, while emphasizing detail depth in logic design topics. Current and future research areas are outlined throughout the course.

Course Objectives:
Students will be exposed to the discipline of electronic design automation, using an analytical approach based on graph-theory and optimization principles. After completing the course, students will be able to describe the flow of VLSI digital logic design, and the algorithms used in various types of design-automation tools. Design representations, algorithms/heuristics and data structures will be practiced in programming assignments. As a result of the course, students will gain insight into EDA, and develop ability to read the professional literature.

Required background:
Digital systems, basic circuit & VLSI concepts.
Programming (in C++ language) and data structures.

Prerequisites:
(044268 Intro. to algorithms and data structures) || (234246/7 Graph Algorithms 1)
Or instructor's consent.

Recommended:
((046237 Intro. to VLSI) || (236354 VLSI Circuit Design) ) &&
((044101 Intro. to Software Systems) || (234122 Intro. to Systems Programming))

Bibliography:
Textbook:

Auxiliary books:
5) Current literature.
**Course plan:**

**Note:**
1) The table below lists multiple options for exercises in class and at home – a subset will be selected each time the course is taught. Lecture material will covered completely.
2) Student will have an average homework load of 4-5 hours per week. For example: turn-in 2 regular homework (H), do 3 programming assignments (P). Homework will be done in groups of 2 students to encourage interaction.

<table>
<thead>
<tr>
<th>week</th>
<th>Lecture topics</th>
<th>Classroom exercise options</th>
<th>Homework assignment options</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><strong>Introduction</strong>&lt;br&gt;Evolution and trends in VLSI CAD. Design process, representations, flow and methodology. CAD Tool types and what they do. VLSI flow overview. Decision and optimization problems in CAD. Example methodology: RTL modeling of synchronous systems.</td>
<td>• Graph-based structural models of hardware&lt;br&gt;• Levelizing algorithm&lt;br&gt;• BFS/DFS comparison</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td><strong>Digital hardware modeling</strong>&lt;br&gt;Spec vs. Description; Structure modeling: Logic networks, connectivity &amp; netlists. Hypergraphs. Connectivity data structures.</td>
<td>• Course connectivity model API&lt;br&gt;• Flattener example</td>
<td>• P1: Connectivity data structure, recursive exploding and flattening</td>
</tr>
<tr>
<td>3</td>
<td><strong>Dynamic verification Simulators</strong>&lt;br&gt;Cycle-based simulation; logic relaxation; event-driven simulator kernels. Value sets. Zero/unit/variable delay simulation &amp; issues. Coverage metrics; instrumentation</td>
<td>• Event queues and scheduling in simulators&lt;br&gt;• Switch level simulation&lt;br&gt;• Comparisons of zero-delay, unit-delay, variable-delay logic simulation</td>
<td>• P2: Compiled code gate level simulation&lt;br&gt;• P3: Event driven interpreted simulation</td>
</tr>
<tr>
<td>4</td>
<td><strong>Dynamic Verification Environments</strong>&lt;br&gt;Limitations: Coverage and speed Higher level models - Transaction Based, Constraint solving Types of coverage: line, expression, flow, event Exceptions/Inline property verification</td>
<td>• Demonstration via example system and verification model&lt;br&gt;• Example “e” constructs&lt;br&gt;• Example TLM</td>
<td>• H2: Complexity of verification environments</td>
</tr>
<tr>
<td>5</td>
<td><strong>Static logic verification</strong> (formal)&lt;br&gt;Advantages of static vs. dynamic verification; Equivalence checking: RTL-to-gates, gatesto-gates, LVS Advanced Boolean algebra: Shannon expansion, cofactors and Unate Recursive computation Paradigm</td>
<td>• Recursive complementation&lt;br&gt;• Properties of cofactors&lt;br&gt;• Equivalence checking with don’t cares</td>
<td>• H3: Working with cube lists: PCN, Cofactors, recursive tautology checking</td>
</tr>
<tr>
<td>6</td>
<td><strong>Binary Decision Diagrams</strong> (BDDs)&lt;br&gt;Representation, properties, variable ordering heuristics. Usage for combinational logic verification;</td>
<td>• Building shared ROBDD&lt;br&gt;• Complemented edges</td>
<td>• H4: Pencil &amp; paper questions on BDDs</td>
</tr>
<tr>
<td>7</td>
<td><strong>BDD operations</strong>&lt;br&gt;Apply, restrict, ITE. BDD forests, BDD drawbacks</td>
<td>• ITE algorithm</td>
<td>• P3: BDD application e.g. Constant propagation or matching nodes</td>
</tr>
<tr>
<td>8</td>
<td><strong>SAT</strong>&lt;br&gt;Principles of SAT solvers, Example problems and application</td>
<td>• SAT application to logic verification or ATPG&lt;br&gt;• Understand minisat</td>
<td>• P4: Gate to Gate Equivalence via minisat&lt;br&gt;• P5: Einstein’s puzzle</td>
</tr>
</tbody>
</table>
|   | Sequential Verification and Formal Model Checking  
Reachability analysis; Symbolic checking using BDDs  
Reduction of logic  
Inferring well know structures | Write properties for example RTL  
Liveness vs. safety | Paper review |
|---|---|---|---|
| 10 | Logic synthesis 1  
2-level minimization: exact vs. Heuristic, Espresso operations; on/off/dc sets; multiple-output functions; Covering problems and algorithms | Espresso algorithms  
Branch and bound algorithm  
Exploiting don’t cares | H5: Using Espresso operations |
| 11 | Logic synthesis 2  
Principles of Multilevel minimization  
Technology mapping :cell libraries; Behavioral synthesis: resource allocation & binding | FSM synthesis  
Propagation of don’t care conditions  
Dynamic programming demo on tech mapping problem | H6: Tech mapping  
H7: theoretical questions about don’t care effects |
| 12 | Static timing analysis 1  
Clocking schemes , sampling elements; definitions of timing parameters and delays; Delay modeling (block/gate/device level); Path analysis algorithms. Min/Max delay verification | Delay modeling, effects of load, waveform slope  
Example of timing design using a path enumeration method | P6: Write interconnect delay estimator  
P7: Write static timing engine given delay graph |
| 13 | Static timing analysis 2  
False paths, sensitization, pessimism, DSM effects: clock skew, noise effects | Static vs. Dynamic Sensitization  
Path ATPG | H8: Questions about path delay algorithms |