Chapter 6 CMOS Design Methods

- Introduction
- VLSI Design Flows
- Design Strategies
- VLSI Design Styles
- Design Verification
System Development Process

- Requirements analysis
- Requirements specification
- Design
- Implementation
- Testing
Design Abstraction

- System layer: How the design interfaces with other systems
- Functional layer: Specification as a block or algorithm
- Circuit layer: Abstract representation
- Layout layer: Fabrication rules and layout
- Physical layer: Semiconductor conduction and electrical properties

Top-down design:

Bottom-up design:
Design Flow – ASIC Design Flow

Start → Design entry → Logic synthesis → System Partitioning → Floorplanning → Placement → Routing

Prelayout simulation → Postlayout simulation → Circuit extraction

Finish
Design Flow – Design Entry

• Enter the design into an ASIC design system, either using a *hardware description language* (HDL) or *schematic entry*

• An example of Verilog HDL

```verilog
module fadder(sum, cout, a, b, ci);
output sum, cout;
input a, b, ci;
reg sum, cout;

always @(a or b or ci) begin
    sum = a^b^ci;
    cout = (a&b)|(b&ci)|(ci&a);
end
endmodule
```
Design Flow – Features of SOC Design

• Parallel development of H/W and S/W
• Parallel verification and synthesis of modules
• Floorplaning, Placement, and Routing in synthesis process
• Use predesigned Macros (soft/hard)
  – Intelligent Property (IP)
• Planned iteration throughput
Design Flow – SOC Design Flow

System Design and Verification

Physical
- Physical specification: area, power, ...
  - Preliminary floorplanning
  - Updated floorplanning
  - Trial placement
- Final placement and routing

Timing
- Timing specification: I/O timing...
  - Block timing specification
  - Block synthesis
  - Top-level synthesis
- Top-level HDL
  - Block verification
  - Top-level verification

Hardware
- Hardware specification: macro decomp...
  - Block selection/design
  - Top-level verification

Software
- Software specification: application development...
  - Application prototype testing
  - Application development
  - Application testing

Time
Design Strategies – Design Parameters

- Performance
  - Speed, power, function, flexibility
- Size of die (cost of die)
- Time to design (cost of engineering and schedule)
- Ease of test generation and testability (cost of engineering and schedule)
Design Strategies – *Hierarchy*

- **Hierarchy**
  - Dividing a module into submodules and then repeating this operation on the submodules

- **Structural hierarchy**
  - Reflect functionality, such as the adding, multiplexing, or storing state

- **Physical hierarchy**
  - An *n*-bit component is built with *n* identical bit-slices
Design Strategies – Example

Structural Hierarchy
Design Styles – Full Custom
Design Styles – Programmable Logic Array

AND array

OR array

Buffering

Inputs

Buffering

Outputs
Design Styles – Gate Array
Design Styles – Standard Cell Design

![Diagram of Standard Cell Design]

- Cell 1
- Cell 2
- Cell 3
- Cell 4
- Cell 5
- Cell 6
- Cell 7
- Cell 8
- Cell 9
- Cell 10
- Cell 11
- Cell 12
- Cell 13
- Cell 14
- Cell 15
- Cell 16
- Cell 17
## Design Styles – Comparison

<table>
<thead>
<tr>
<th>Design Styles</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Full-custom</strong></td>
<td>- Compact designs;</td>
<td>- Very time consuming;</td>
</tr>
<tr>
<td></td>
<td>- Improved electrical characteristics;</td>
<td>- More error prone;</td>
</tr>
<tr>
<td><strong>Semi-custom</strong></td>
<td>- Well-tested standard cells which can be shared between users;</td>
<td>- Can be time consuming to built-up standard cells;</td>
</tr>
<tr>
<td></td>
<td>- Good for bottom-up design;</td>
<td>- Expensive in the short term but cheaper in long-term costs;</td>
</tr>
<tr>
<td><strong>Gate array</strong></td>
<td>- Fast implementation;</td>
<td>- Can be wasteful of space and pin connections;</td>
</tr>
<tr>
<td></td>
<td>- Easy updates;</td>
<td>- Relatively expensive in large volumes;</td>
</tr>
<tr>
<td></td>
<td>- Only two layers of metal require customization;</td>
<td></td>
</tr>
</tbody>
</table>

National Central University EE613 VLSI Design
Design Verification – Basic Concept

• A conventional flow through a set of design tools to produce a CMOS chip from a functional specification
Design Verification – Simulation

• Circuit-level simulation – SPICE
  – High accuracy
  – Long simulation times
  – Basic sources of error
    * Inaccuracies in the MOS model parameters
    * An inappropriate MOS model
    * Inaccuracies in parasitic capacitances and resistances

• Logic-level simulation
  – Deal with simulation at the logic level
  – Timing
    * Specified with an intrinsic delay and a load dependent delay
Design Verification – *Simulation*

- An event-driven simulator
- Timing is specified with $T_{\text{gate}} = T_{\text{intrinsic}} + C_{\text{load}} \times T_{\text{load}}$

**Switch-level simulation**
- Switch simulators merge logic-simulator techniques with some circuit simulation techniques by modeling transistors as switches
- Modeling CMOS gates as either pull-up or pull-down structures

**Mixed-mode simulators**
- Simulators that merge the good points of functional simulation, logic simulation, switch simulation, timing simulation, and circuit simulation
Design Verification – Summary

• A good simulator is crucial to modern CMOS design

• Logic simulators are of use at the system level

• Timing simulators are useful for modules into the 100-100K transistors

• Circuit simulators are useful for 10-1000 transistors

• Mixed-mode simulators allow a trade-off in simulation accuracy and time of simulation