VLSI Design Methodology

- The design process then and now
- Design abstraction
- Role of EDA tools
- Reusable designs
- Today’s role of VLSI design engineer
- VLSI Facts of Life
Evolution in Complexity

[Graph showing the evolution of logic IC gates and DRAM bits over time, with trend lines and labels indicating improvements every 5 years.]
Evolution in Transistor Count
The Design Process - 25 Years Ago

Intel 4004 Micro-Processor

• NMOS only process

• Truly “handcrafted”

• Each transistor laid out and optimized individually

• Layouts done by specialists working with highly magnified drawings

• No automated verification or simulation tools available
The Design Process - Today

• Typically contain more than 1 million transistors
• Time to market measured in months (not years)
• Rapidly evolving technologies (technology scales every two years)
  • Presently 0.18u channel widths; 0.06u by 2010?
The Design Process - Today

Example:

Intel Pentium

• CMOS technology
• ~3.5M transistors
• Circuit is hierarchical
• Cells reused as much as possible
Design Abstraction

Highest level of abstraction (RTL)

Processor

Characterized by behavioral model *Adder, Decoder, Mux*

Logic function (Boolean eqns)

Switch (QuickSim II)

SPICE model (Accusim)
CAD Tools (EDA)

As designs get more complex, we really need machines to make machines.

- organize
- generate
- verify

- Standard-cell place and route for “random” logic.
- Symbolic layout tools to ease the task of physical design; mask verification to ensure manufacturability.

- Circuit analysis programs predict circuit behavior at all the process corners. Gate-level and behavioral simulators help you get it right the first time!

- Tools to do the tedious, repetitive work such as routing, “tiling” a mosaic of building-block cells, or verifying that the layout and schematic match.
Reusing Designs

Q: How to avoid reinventing the wheel each time new chip is designed?

Cell Libraries:

• Avoid redesign and reverification of frequently used cells (basic gates, arithmetic and memory modules)

• Contain the layout of the individual cells

• Provide complete documentation and characterizes behavior of the cell

Make a Cell Library!
Circuit Design and Layout

Standard Cell

- Each cell has same height
- Cells are abutted to form rows, and connected together
- Cells share common supply rails
- Space between rows used for routing
Circuit Design and Layout

Full Custom

• Layout optimized by hand

• Optimized blocks can be tiled to make bigger blocks

• Most efficient layout scheme

• Takes longer to verify (analog simulation on whole block)

• Very tedious

EE415 VLSI Design
Circuit Design and Layout

RAM Generator

• Optimized layout generated automatically

• Any size layout can be generated (parameterizable)

• Works only for regular structures (RAM)
Circuit Design and Layout

Which engineer drew the most number of transistors?

• Full Custom
If EDA solves the problem, why need engineers?

- Someone still has to design and implement the cell libraries
  - “porting” cells to new technologies not foolproof
  - technology changes every two years (approx.)
- Identify critical timing paths to develop accurate model
- Library approach good enough for ASICs; not for high performance
  - main goal of ASIC is integration
  - ASIC requirements fit easily within capability of technology
  - designs are “tweaked” to the max for high-performance designs (microprocessors, DSP)
  - No substitute for human ingenuity
If EDA solves the problem, why need engineers?

• Abstraction-based approach has its limits
  • performance of modules are seriously affected by interconnects
  • interconnects parasitic effects increase as technologies are scaled down
  • effect is the most profound on global signals
    • clock distribution
    • circuit synchronization
    • supply-voltage distribution
• New design issues and constraints
  • power dissipation limits
• Murphy’s Law!
  • Troubleshooting requires expertise
VLSI: The Ideal Implementation Medium?

So, what not to like about VLSI?

• Gives the designer control over almost all aspects of the design: architecture, logic choice, speed, area, power, ...
• Densities are increasing, costs decreasing with each passing year
• Is used by almost everyone (fabless or fab): “No one gets fired for building an ASIC”
• Created technology responsible for much of the economic growth of the 80’s and 90’s. It will no doubt continue in its starring role for some years to come.

Is life really a bowl of cherries?
VLSI fact-of-life #1: “So much to do, so little time”

What you need is a design methodology:

- Budget ($, speed, area, power, schedule, risk)
- high-level architecture, Low-level building blocks
- Behavioral design, verification
- Logic design, verification
- Layout, verification
VLSI fact-of-life #2: “You can’t reach in and fix it”

Once the chip is made, that’s it; chip cannot be repaired.
The word “verification” kept appearing in the previous slide

Mistakes can be costly:

<table>
<thead>
<tr>
<th>Time</th>
<th>Money</th>
</tr>
</thead>
<tbody>
<tr>
<td>Find bug(s) as you go</td>
<td>?</td>
</tr>
<tr>
<td>Reverify 1wk</td>
<td>$10k</td>
</tr>
<tr>
<td>New masks 3days</td>
<td>$1k/wafer</td>
</tr>
<tr>
<td>Slip ship date</td>
<td>$$$$$$$</td>
</tr>
</tbody>
</table>
VLSI fact-of-life #2: “You can’t reach in and fix it”

There’s a lot that needs checking:

• chip has to be manufacturable
  > verified at mask level

• circuit must operate at all “corners”
  > verified at building block level

• logic must be correct
  > verified at RTL/gate level

• chip has to interoperate with system
  > verified at behavioral level
VLSI fact-of-life #3: “You can’t find all the bugs”

The key word here is “find”:

• one can’t explore the circuit under all possible working conditions
• some bugs arise from interactions that one never thinks of
• one can spend forever looking for bugs! Time pressures mean that most searches stop too soon

The trick is to choose implementation rules resulting in a correct by construction design (at least avoids as many problems as possible)

For example:

• choose a simple clocking scheme
• module inputs must go to FET gates only
• use poly only for local interconnect
• no diffusion wires, etc.
VLSI fact-of-life #4: “Nobody’s perfect”

Plan for what to do after you turn it on and nothing happens

- Provide lot’s of observability and controllability. You’ll need to localize and then fix the bug.
- Have a way to run the chip slowly and/or stop it without it burning up or loosing bits.
- Figure out how to check performance without relying on fast I/O (tester pins are slow!)
- Leave room in the budget (time, $) for debugging
- Write and run your manufacturing tests before tape out