CSE241
VLSI Digital Circuits
Winter 2003

Lecture 13: Verification
Question 1: Clocking and Place-Route Flow. Consider the following steps:

- Clock sink placement
- Standard-cell global placement
- Standard-cell detailed placement
- Standard-cell ECO placement
- Clock buffer tree construction
- Global signal routing
- Detailed signal routing
- Bounded-skew (balanced) clock (sub)net routing
- Steiner clock (sub)net routing
- Clock sink useful skew scheduling (i.e., solving the linear program, etc.)
- Post-placement (global routing based) static timing analysis
- Post-detailed routing static timing analysis

(A) As the designer of a clock distribution flow for high-performance standard-cell based ASICs, how would you use and order these steps? It is possible to use some steps more than once, others not at all (e.g., if subsumed by other steps).

(B) List the criteria you used for assessing possible flows. (C) What were the 3 next-best flows that you considered (describe as variants of your flow), and explain why you prefer your given answer. (10 points)
Takehome Midterm Questions

Question 2: Scaling of Dynamic Clock Power. As we move from one technology node to the next, suppose that:

- Minimum feature size ($L_{\text{gate}}$) is scaled by 0.7x
- All metal layers (width, pitch) scale with minimum feature size except for $M_{\text{top}}$ which is unscaled
- Supply voltage is scaled by 0.85x
- Clock frequency is scaled by 2x
- Wire thickness remains the same for $M_{\text{top}}$, $M_{x}$, and $M_{1}$, but the number of metal layers increases by 1
- The number of clock sinks in a given-size (say, 1cm²) die doubles
- The maximum number of loads driven by any given buffer in the clock buffer tree remains constant at 4, and somehow all buffers are the same multiple (e.g., PMOS W/L = 80) of minimum-size
- The clock is distributed by a perfectly balanced, buffered H-tree (so, sinks are placed on a perfect grid, etc.)

Give your analysis of how dynamic power dissipation due to clock distribution scales every TWO technology nodes, for a constant-size die. Note any additional assumptions needed. You can break up your analysis into different components (wire capacitance, gate capacitance, etc.) as you find appropriate. (10 points)

Question 3: Design Data Volume. If we migrate to the next technology node and double the gate count of a design, how would you expect the size of the LEF and routed DEF files to change? Explain your reasoning. (5 points)
Homework Solutions

Q4. Coupling can be controlled by:

- Introduction of shield lines that are tied to power or ground supplies; these give stable MCF = 1 to nearest-neighbors (no delay uncertainty due to capacitive coupling) and also closer current return paths (less loop area \(\rightarrow\) less inductance). Dedicated ground and power planes are also possible.

- Increased spacing between lines (reduces capacitive coupling).

- Control of slew times to be between prescribed lower and upper bounds; too-short slew times create strong aggressor lines, while too-long slew times create weak victim lines.

- Sizing of gates to juggle drive strengths and load capacitances, in order to control slew times.

- Staggering of repeaters so that worst-case MCF can occur only along one half of the repeater distance (= period between consecutive repeaters), and if it does, best-case MCF will occur along the other half.

- Swizzling, e.g., of parallel bus wires can prevent long parallel runs of neighboring interconnects; sometimes this is effectively achieved by “randomization” of the routing (a feature, not a bug).

- Inductance can also cause L \(\frac{di}{dt}\) supply noise, and this can be controlled by power/ground network design, location and assignment of power/ground pins, explicitly added decoupling capacitances, and increasing slew times.
Q5. If we assume that the R’s and C’s are for interconnect segments between nodes that are not explicitly shown, and we use the expression given in the slide:

\[
T_{ED}(\text{In},\text{O}_2) = R_1 \left( \frac{C_1}{2} + C_2 + C_3 + C_4 + C_5 + C_6 \right) \\
+ R_4 \left( \frac{C_4}{2} + C_5 + C_6 \right) \\
+ R_5 \left( \frac{C_5}{2} + C_6 \right) \\
+ R_6 \left( \frac{C_6}{2} \right)
\]

Evaluation of Elmore delay requires time linear in the size of the RC tree. With one depth-first traversal of the tree, we can obtain the total downstream capacitance (i.e., subtree capacitance) at every node of the tree; this value is correctly captured the last time any given node is visited in the DFS. Then, a second depth-first tree traversal yields the delay from the source to every node of the tree; this value is correctly captured the first time any given node is visited in the DFS.

Q6. We have \( C_L = xC_{in} = U^N C_{in} \) and total propagation delay \( t_p = NUt_0 \) where \( t_0 \) is the delay of a minimum-size inverter driving another minimum-size inverter as its fanout load. Then, \#stages * delay per stage gives

\[
t_p = \ln X / \ln U \times Ut_0
\]

and setting \( \frac{dt_p}{dU} = \ln X ((\ln U - U * 1/U) / \ln U) = 0 \) yields \( \ln U - 1 = 0 \rightarrow U = e \)
This Class + Logistics

- Verification (functional, physical)

- Schedule
  - Lecture #14: Manufacturing

  - Expect email from anuj@vlsicad.ucsd.edu with pointer to pdf (scanned pp. 109-156).

- Lab #5 due Wednesday, Lab #6 (Formality) on Wednesday.

- Project revisions due to runtimes (?)
Functional Verification
Design Verification

Is design consistent with original spec?
Implementation Verification

Is implementation consistent with original design intent?
Manufacture Verification (Test)

Is manufactured circuit consistent with implemented design?
Apply gate-level simulation at each step to verify:

1. **Functionality:**
   - 0-1 behavior on regression test set

2. **Timing:**
   - Maximum delay of circuit on critical paths
Advantages of gate-level simulation

- verifies timing and functionality simultaneously
- approach well understood by designers

Disadvantages of gate-level simulation?

- computationally intensive - only 1 - 10 clock cycles of 100K gate design per 1 CPU second
- incomplete - results only as good as your vector set - easy to overlook incorrect timing/behavior
Alternative - Static Signoff

Use static analysis techniques to verify:

1. **Functionality**: formal equivalence-checking techniques
2. **Timing**: static timing analysis

---

**Library/module generators**

- HDL
  - RTL Synthesis
    - netlist
      - logic optimization
        - netlist
          - physical design
            - layout

---

**ASiC signoff**

Courtesy K. Keutzer, UCB

A. B. Kahng, UCSD 2003
Problem: RTL to RTL Verification

- After verification RTL may still be modified for:
  - performance
  - power
  - area
  - testability

- Need to verify that new RTL is correct
Problem: RTL to Gates Verification

- Verify the gate level implementation is consistent with the RTL level design.
- Errors may have occurred due to
  - Logic synthesis
  - Manual intervention

HDL Design  Implementation

Courtesy K. Keutzer, UCB
Problem: Gates to Gates Verification

- Verify the modified gate level implementation is consistent with the RTL level design.

- Errors may have occurred due to:
  - Incorrect logic synthesis or module generation
  - Test insertion
  - Scan chain reordering
  - Clock tree synthesis
  - Post layout “tweaks”
Problem: Layout to Gates Verification (LVS)

- Verify that modified gate level implementation is consistent with the RTL level design

- Errors may have occurred due to
  - Errors in physical design tools
  - Manual changes in layout

- Verification is primarily graphical or "topological"

![Diagram showing netlist and physical layout with an arrow indicating the verification process.](image-url)
Solving Layout to Gates Verification (LVS)

- Extract gate level models from physical level
- Graphically compare extracted model against gate-level schematic (layout versus schematic)
- Flag any discrepancies
Solving Gates to Gates Verification

“specification”

implementation
Extract Combinational Portions

``spec''

Flip-flops

Combinational Logic

inputs

outputs

``implementation''

Flip-flops

Combinational Logic

inputs

outputs

compare combinational portions

Courtesy K. Keutzer, UCB
Combinational Equivalence Checking

- Presumes equivalence relation given (or discovered) between sequential circuits

- Approaches
  - Canonical forms - BDDs and variants
  - Test-oriented methods
  - Symbolic manipulation
    - graph isomorphism
    - structural reductions

- These techniques form the foundation of implementation verification
The Comparison Mitre

Primary Inputs, Register and Black Box Outputs

spec

implementation

COMPARE

0 or 1

Courtesy K. Keutzer, UCB
Verification and Testing

- Given two single-output circuits A and B
- Are A and B equivalent can be posed as: Is there a test for \( F_{s-a-0} \)?

- If \( F_{s-a-0} \) is redundant, \( A \equiv B \) else test vector produces different outputs for A and B.

\[ \x_1 \x_2 \x_3 \x_4 \]

\[ A \]

\[ B \]

\[ s-a-0 \]
Enumerate the ON-set cubes, i.e., all tests for s-a-0 on A. Simulate these tests on B to check if B produces a 1. If not A \( \not\equiv \) B.

Do for OFF-set of A as well.
Reduction: Solving RTL-to-Gates Verification

Step 1: (formally) translate HDL source into netlist

Step 2: Perform gates-to-gates verification

Combinational logic

Combinational logic

Combinational logic

HDL "specification"

RTL Synthesis

netlist

Courtesy K. Keutzer, UCB
Reduction: Solving RTL-to-RTL Verification

Step 1: (formally) translate both HDL sources into netlists

Step 2: perform gate-to-gate verification on netlists
Equivalence Checking Summary

- Routinely verify complex (>1M gate) integrated circuit designs
- Commercial (e.g. Synopsys 44% market share, Verplex 33%) and proprietary (e.g. IBM) solutions exist
- Static sign-off methodology more widely used
- Successful equivalence checkers orchestrate several different approaches
  - syntactic equivalence
  - automatic test pattern generation-like approaches
  - BDD-based techniques
  - pattern-reduction methods
- Open issues
  - retimed circuits
  - circuits with differing state assignments
Retimed Circuits

Circuits are equivalent (modulo some initial state issues) but it is not possible to show that they are equivalent using Boolean equivalence

Courtesy K. Keutzer, UCB
Encoding Issues

- Some logic specifications are “symbolic” rather than binary-valued
  e.g. specification for an ALU

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD</td>
<td>+</td>
</tr>
<tr>
<td>SUB</td>
<td>-</td>
</tr>
<tr>
<td>XOR</td>
<td>Exclusive-OR</td>
</tr>
<tr>
<td>INC</td>
<td>Increment</td>
</tr>
</tbody>
</table>

- Can assign any binary code to the symbolic values, so long as they are different

Courtesy K. Keutzer, UCB
### Different Encodings

<table>
<thead>
<tr>
<th>Circuit 1</th>
<th>Symbol</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD</td>
<td></td>
<td>00</td>
</tr>
<tr>
<td>SUB</td>
<td></td>
<td>01</td>
</tr>
<tr>
<td>XOR</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>INC</td>
<td></td>
<td>11</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Circuit 2</th>
<th>Symbol</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD</td>
<td></td>
<td>11</td>
</tr>
<tr>
<td>SUB</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>XOR</td>
<td></td>
<td>00</td>
</tr>
<tr>
<td>INC</td>
<td></td>
<td>01</td>
</tr>
</tbody>
</table>

Different encodings make circuits no longer amenable to combinational logic equivalence checking.
Physical Verification
Overview

What is Physical Verification (PV)?

General PV concepts
- Design Rule Check (DRC)
- Logical Versus Schematic (LVS)
- Verification Algorithms
  - Flat and Hierarchical

Approaches
- DRC
  - Place and Route, Flat and Hierarchical
- LVS
  - Place and Route, Flat and Hierarchical
Design Rule Checks (DRCs)

Goals:
- Manufacturability
- Yield

Analysis Inputs:
- Foundry
  - Rules
- Design data
  - Mask data, Layer information

Typical checks performed:

For Manufacturing
- Width, Spacing, Minimum Area, Enclosed Area, Overhang, etc.

For Yield
- Antenna, Electromigration,Latch-up, Electrostatic Discharge, Density
Design Rule Waivers

- Well tested special structures
  - Memory macros

- Special permissions with the cost of reduced yield
  - Antenna rules
  - Density rules
  - EM rules
Layout Versus Schematic (LVS)

Goals:
- Functionality

Analysis Inputs:
- Foundary or Library Vendor
  - Library Spice Netlist
- Design data
  - Mask data, Logic Netlist

Typical checks performed:
- Connectivity Recognition
- Device Recognition

![Diagram showing LVS process]

Spice Netlist ➔ Design (Layout & Netlist) ➔ LVS ➔ Violations Markers ➔ Violations Report

Courtesy Cadence Design Systems, Inc.

A. B. Kahng, UCSD 2003
Analysis Process

Steps:
- From Gates to Transistors
- Primary I/Os Identified
- Connectivity Traced
- Device Recognition

Design Netlist:

Design Layout:

Design Transistors:

Courtesy Cadence Design Systems, Inc.
Flat Verification

All Levels Flattened to a Single Level

Verification Performed on a Flat Database

Courtesy Cadence Design Systems, Inc.
Hierarchical Verification

- Layout Netlist
- Schematic Netlist
- Top Level Verified
- Cells Verified
- Cells Verified
- Cells Verified

Check

- T1
- H1
- H2
- C1
- C2

Cells Verified
Cells Verified
Cells Verified

Skip

Cells not verified since C1 already checked

Courtesy Cadence Design Systems, Inc.

A. B. Kahng, UCSD 2003
Approaches

- **DRC**
  - Place and Route Environment
  - Flat
  - Hierarchical

- **LVS**
  - Place and Route Environment
  - Flat
  - Hierarchical
DRC: In Place and Route

Description:
All cells are modeled with abstract. No layout available.

Advantages:
- Fast
- Small database
- Problems can be debugged and fixed fast.

Disadvantages:
- Checking as accurate as abstracts.
- No checks at different hierarchy levels.
- Connection to pins could have violations.
DRC: Flat

Description:
All cells are flattened. All geometric shapes visible. No black boxes.

Advantages:
- Single run for entire chip, simple to setup
- Has to be performed prior to every tape out
- No modeling requirements

Disadvantages:
- Entire design completed
- Long run times
- Resource requirements
- Harder to debug

Sub blocks merged at the top
Hard Macro Flattening
IO Pad Flattening

Courtesy Cadence Design Systems, Inc.
DRC: Hierarchical

Description:
- Bottom-up checking starting at block/hard macro level
- Blocks verified separately
- Top level verified using black box models for sub-blocks

Advantages:
- Start before entire chip completed
- Smaller data size = shorter run times, simpler debugging, easier to fix
- Early density, EM, wide metal checks and repair
- Effects seen on timing, SI early when it can still be addressed

Disadvantages:
- Proper modeling of over the block and through the block routes
- Full flat chip analysis is still required
- Density checks may be inaccurate
- Assumptions made at hierarchy boundaries
LVS: In Place and Route

Description:
All cells are modeled with abstract. No cell layout and netlist available.

Advantages:
- Fast
- Small database
- Problems can be debugged and fixed fast.

Disadvantages:
- Only connectivity check of the nets.
- No checks at different hierarchy levels.

Sub blocks modeled with abstracts
Hard Macro modeled with abstracts
IO Pad modeled with abstracts

Courtesy Cadence Design Systems, Inc.
**LVS: Flat**

**Description:**
- Design flattened to one level.
- Primary I/Os and supply I/Os labeled.
- Entire IC layout compared to transistor level schematic.

**Advantages:**
- Simple setup, implementation.
- No modeling requirement.
- Run before all tape outs regardless.

**Disadvantages:**
- Large data yielding long run times.
- Hard to debugging.
- Late in design cycle hard to accommodate changes.
LVS: Hierarchical

Description:
- Bottom-up checking starting at block/hard macro level
- Blocks verified separately
- Top level verified with black box models for sub-blocks
- Connections to black boxes checked but not content

Advantages:
- Reduced amount of data yielding faster run times
- Easier to debug
- Data maturity (incomplete block)
- Fixing problems early in design easier
- IP issues, verification reuse

Disadvantages:
- Full flat chip analysis still required
- Modeling errors possible
Hierarchical Filling Problem

- Filling geometries are added only to master cells.
- Each cell of the filled layout is a filled version of the corresponding original master cell.
Why Hierarchical Filling?

- Hierarchy characteristics of custom and semi-custom design flows
- Enables and faster verification of the filled layout
- Decreases data volume for standard cell designs
Difficulties of Hierarchical Filling

- Density constraints for all instances of the master
- Interactions / interferences at master cell boundaries
- Always worse than flat solutions
**k-way Master Cell Splitting**

- Create \( k \) copies of master cell \( C_i \)
- Link all contained master cell \( C' \) with the new copies of \( C_i \)
- Randomly replace \( C_i \) in any master cell with one of the new copies

\[ C_i,1 \rightarrow C_i,2 \]

- \( k \rightarrow \infty \): hierarchical layout \( \rightarrow \) flat layout
Hybrid Hierarchical / Flat Filling

- Purely hierarchical fill phase
- Split-hierarchical phase
- Flat fill `cleanup` phase

Features:
- three instances of a master cell
Physical Verification Summary

- Tool modes
  - Hierarchical vs. Flat

- Examining DRC and LVS errors
  - Design rule waivers

- DRC and LVS approaches
  - Place and Route
  - Flat
  - Hierarchical

- Dummy fill insertion
  - Flat
  - Hierarchical