Introduction to CMOS VLSI Design

Slides adapted from:

N. Weste, D. Harris, CMOS VLSI Design, © Addison-Wesley, 3/e, 2004

Outline

- A Brief History
- MOS transistors
- CMOS Logic
- CMOS Fabrication and Layout
- Design Flow
  - System Design
  - Logic Design
  - Physical Design
  - Design Verification
  - Fabrication, Packaging and Testing
CMOS Fabrication

- CMOS transistors are fabricated on a thin silicon wafer that serve as both a mechanical support and electrical common point called substrate.
- Lithography process similar to printing press.
- On each step, different materials are deposited or etched.
- Easiest to understand by viewing both top and cross-section of wafer in a simplified manufacturing process.

Inverter Cross-section

- Typically use p-type substrate for nMOS transistors.
- Requires n-well for body of pMOS transistors.
Well and Substrate Taps

- Substrate must be tied to GND and n-well to V_{DD}.
- Metal to lightly-doped semiconductor forms poor connection called Shottky Diode.
- Use heavily doped well and substrate contacts / taps.

Inverter Mask Set

- Transistors and wires are defined by masks.
- Cross-section taken along dashed line.
Detailed Mask Views

- Six masks
  - n-well
  - Polysilicon
  - n+ diffusion
  - p+ diffusion
  - Contact
  - Metal

Fabrication Steps

- Start with blank wafer
- Build inverter from the bottom up
- First step will be to form the n-well
  - Cover wafer with protective layer of SiO₂ (oxide)
  - Remove layer where n-well should be built
  - Implant or diffuse n dopants into exposed wafer
  - Strip off SiO₂
Photoresist

- Spin on photoresist
  - Photoresist is a light-sensitive organic polymer
  - Softens where exposed to light

Lithography

- Expose photoresist through n-well mask
- Strip off exposed photoresist
Etch

- Etch oxide with hydrofluoric acid (HF)
  - Seeps through skin and eats bone; nasty stuff!!!
  - Only attacks oxide where resist has been exposed

n-well

- n-well is formed with diffusion or ion implantation
- Diffusion
  - Place wafer in furnace with arsenic gas
  - Heat until As atoms diffuse into exposed Si
- Ion Implantation
  - Blast wafer with beam of As ions
  - Ions blocked by SiO₂, only enter exposed Si
Strip Oxide

- Strip off the remaining oxide using HF
- Back to bare wafer with n-well
- Subsequent steps involve similar series of steps

Polysilicon

- Deposit very thin layer of gate oxide
  - < 20 Å (6-7 atomic layers)
- Chemical Vapor Deposition (CVD) of silicon layer
  - Place wafer in furnace with Silane gas (SiH₄)
  - Forms many small crystals called polysilicon
  - Heavily doped to be good conductor
Polysilicon Patterning

- Use same lithography process to pattern polysilicon

Self-Aligned Process

- Use oxide and masking to expose where n+ dopants should be diffused or implanted
- N-diffusion forms nMOS source, drain, and n-well contact
N-diffusion

- Pattern oxide and form n+ regions
- **Self-aligned process** where gate blocks diffusion
- Polysilicon is better than metal for self-aligned gates because it doesn’t melt during later processing

N-diffusion cont.

- Historically dopants were diffused
- Usually ion implantation today
- But regions are still called diffusion
N-diffusion cont.

- Strip off oxide to complete patterning step

P-Diffusion

- Similar set of steps form p+ diffusion regions for pMOS source and drain and substrate contact
Contacts

- Now we need to wire together the devices
- Cover chip with thick field oxide
- Etch oxide where contact cuts are needed

Metalization

- Sputter on aluminum over whole wafer, filling the contacts as well
- Pattern to remove excess metal, leaving wires
Layout

- Chips are specified with set of masks
- Minimum dimensions of masks determine transistor size (and hence speed, cost, and power)
- Feature size $f =$ distance between source and drain
  - Set by minimum width of polysilicon
- Feature size improves 30% every 3 years or so
- Normalize for feature size when describing design rules
- Express rules in terms of $\lambda = f/2$
  - E.g. $\lambda = 0.3$ $\mu$m in 0.6 $\mu$m process

Layout Design Rules

Conservative rules to get started!

Fig 1.39 Simplified $\lambda$-based design rules for layouts with 2-metal layers (MOSIS)
Design Rules Summary

- Metal and diffusion have minimum width and spacing of 4\(\lambda\).
- Contacts are 2\(\lambda\) x 2\(\lambda\) and must be surrounded by 1\(\lambda\) on the layers above and below.
- Polysilicon uses a width of 2\(\lambda\).
- Polysilicon overlaps diffusions by 2\(\lambda\) where a transistor is desired and has spacing or 1\(\lambda\) away where no transistor is desired.
- Polysilicon and contacts have a spacing of 3\(\lambda\) from other polysilicon or contacts.
- N-well surrounds pMOS transistors by 6\(\lambda\) and avoid nMOS transistors by 6\(\lambda\).

Gate Layout

- Layout can be very time consuming
  - Design gates to fit together nicely
  - Build a library of standard cells
- Standard cell design methodology
  - \(V_{DD}\) and GND should abut (standard height)
  - Adjacent gates should satisfy design rules
  - nMOS at bottom and pMOS at top
  - All gates include well and substrate contacts
Transistor dimensions specified as W / L ratio
- Minimum size is $4\lambda / 2\lambda$, sometimes called 1 unit
- In $f = 0.6 \mu m$ process, this is 1.2 $\mu m$ wide, 0.6 $\mu m$ long

Example: Inverter Standard Cell Layout
Example: 3-input NAND Standard Cell Layout

- Horizontal n-diffusion and p-diffusion strips
- Vertical polysilicon gates
- Metal1 VDD rail at top
- Metal1 GND rail at bottom
- 32 $\lambda$ by 40 $\lambda$

Stick Diagrams

- **Stick diagrams** help plan layout quickly
  - Need not be to scale
  - Draw with color pencils or dry-erase markers
Wiring Tracks

- A **wiring track** is the space required for a wire
- $4\lambda$ width, $4\lambda$ spacing from neighbor = $8\lambda$ pitch
- Transistors also consume one wiring track

Well spacing

- Wells must surround transistors by $6\lambda$.
  - Implies $12\lambda$ between opposite transistor flavors
  - Leaves room for one wire track
Area Estimation

- Estimate area by counting wiring tracks
  - Multiply by 8 to express in $\lambda$.

Design Challenges

- The greatest challenge in modern VLSI is not in designing the individual transistors but in managing system complexity.
- Modern System-on-Chip designs use:
  - Many millions (soon billions!) of transistors
  - Tens to hundreds of engineers
- How to cope with Complexity?
  - Abstraction Level
  - Structured Design
  - Design Flow
Design Abstraction Levels

Structured Design Tenets

- Hierarchy
  - Divide and Conquer
- Regularity
  - Reuse modules wherever possible
  - Ex: standard cell library
- Modularity
  - Well-formed interface allows modules to be treated as black boxes
- Locality
  - Physical and Temporal
Design Flow

- **Architecture**: User’s perspective, what does it do?
  - Instruction set, MIPS, x86, Alpha, PIC, ARM, ...
- **Microarchitecture**
  - Single cycle, multi cycle, pipelined, superscalar?
- **Logic**: how are functional blocks constructed?
  - Ripple carry, carry look ahead, carry select adders
- **Circuit**: how are transistors used?
  - Static CMOS, pass transistors, domino logic, ...
- **Physical**: chip layout
  - Datapaths, memories, random logic

Design Flow: Gajski’s Y-Diagram

The design process can be viewed as making transformations from one domain to another while maintaining the equivalency of the domains.

Y Diagram: Reproduced from [King03] with permission of The McGraw-Hill Companies.
Architecture: MIPS example

- Instruction Set
-Instruction encoding formats

**Table 1.7 MIPS instruction set (subset supported)**

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Function</th>
<th>Encoding</th>
<th>op</th>
<th>funct</th>
</tr>
</thead>
<tbody>
<tr>
<td>add $1, $2, $3</td>
<td>$1 &lt;- $2 + $3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>sub $1, $2, $3</td>
<td>$1 &lt;- $2 - $3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>and $1, $2, $3</td>
<td>$1 &lt;- $2 and $3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>or $1, $2, $3</td>
<td>$1 &lt;- $2 or $3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>sll $1, $2, $3</td>
<td>set less than: $1 &lt;- 1 if $2 &lt; $3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>add $1, $0, imm</td>
<td>add immediate: $1 &lt;- $0 + imm</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>beq $1, $2, imm</td>
<td>branch if equal: $1 &lt;- $2 + imm</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>j destination</td>
<td>jump: PC = destination</td>
<td>J</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>lb $1, imm($2)</td>
<td>load byte: $1 &lt;- mem($2 + imm)</td>
<td>I</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>sb $1, imm($2)</td>
<td>store byte: mem($2 + imm) &lt;- $1</td>
<td>I</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

a. Technically, MIPS addresses specify bytes. Instructions require a four-byte word and must begin at addresses that are a multiple of four. To most effectively use instruction bits in the full 32-bit MIPS architecture, branch and jump constants are specified in words and must be multiplied by four (shifted left two bits) to be converted to byte addresses.

Architecture: MIPS example cont.

**Table 1.49 Instruction encoding formats**

<table>
<thead>
<tr>
<th>Format</th>
<th>Example</th>
<th>Encoding</th>
<th>dest</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>add $rd, $ra, $rb</td>
<td>0 ra rb rd 0 funct</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>6 5 5 5 5 6</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>beq $ra, $rb, imm</td>
<td>op ra rb imm</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>6 5 5</td>
<td>16</td>
</tr>
<tr>
<td>J</td>
<td>j destination</td>
<td>op dest</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>6 26</td>
<td></td>
</tr>
</tbody>
</table>
Logic Design: MIPS example

- Start at the top level
  - Define the top-level interface
  - Top level block diagram
  - Hierarchically decompose top level into units
  - Design the units (e.g. HDL)

FIG 1.55  MIPS computer system
Logic Design: MIPS top level block diagram

Logic Design: MIPS Hierarchy
Logic Design using HDLs

- Hardware Description Languages
  - Widely used in logic design
  - Verilog and VHDL
- Describe hardware using code
  - Document logic functions
  - Simulate logic before building
  - Synthesize code into gates and layout
    - Requires a library of standard cells

Circuit Design

- How should logic be implemented?
  - NANDs and NORs vs. ANDs and ORs?
  - Fan-in and fan-out?
  - How wide should transistors be?
- These choices affect speed, area, power
- Logic synthesis makes these choices for you
  - Good enough for many applications
  - But when a system has critical requirement
    Hand-crafted circuits are still better
Example: Full Adder

Example: Carry circuit

- **VERILOG:**
  
  ```verilog
  assign cout = (a&b) | (a&c) | (b&c);
  ```

FIG 1.58 Full adder

FIG 1.59 Carry subcircuit
Gate-level Netlist

module carry(input a, b, c, output cout)

wire x, y, z;

and g1(x, a, b);
and g2(y, a, c);
and g3(z, b, c);
or g4(cout, x, y, z);

endmodule

Transistor-Level Netlist

module carry(input a, b, c, output cout)

wire i1, i2, i3, i4, cn;

tranif1 n1(i1, 0, a);
tranif1 n2(i1, 0, b);
tranif1 n3(cn, i1, c);
tranif1 n4(i2, 0, b);
tranif1 n5(cn, i2, a);
tranif0 p1(i3, 1, a);
tranif0 p2(i3, 1, b);
tranif0 p3(cn, i3, c);
tranif0 p4(i4, 1, b);
tranif0 p5(cn, i4, a);
tranif1 n6(cout, 0, cn);
tranif0 p6(cout, 1, cn);

endmodule
SPICE Netlist

```
.SUBCKT CARRY A B C COUT VDD GND
MN1 I1 A GND GND NMOS W=1U L=0.18U AD=0.3P AS=0.5P
MN2 I1 B GND GND NMOS W=1U L=0.18U AD=0.3P AS=0.5P
MN3 CN C I1 GND NMOS W=1U L=0.18U AD=0.5P AS=0.5P
MN4 I2 B GND GND NMOS W=1U L=0.18U AD=0.15P AS=0.5P
MN5 CN A I2 GND NMOS W=1U L=0.18U AD=0.5P AS=0.15P
MP1 I3 A VDD VDD PMOS W=2U L=0.18U AD=0.6P AS=1P
MP2 I3 B VDD VDD PMOS W=2U L=0.18U AD=0.6P AS=1P
MP3 CN C I3 VDD PMOS W=2U L=0.18U AD=1P AS=1P
MP4 I4 B VDD VDD PMOS W=2U L=0.18U AD=0.3P AS=1P
MP5 CN A I4 VDD PMOS W=2U L=0.18U AD=1P AS=0.3P
MN6 COUT CN GND GND NMOS W=2U L=0.18U AD=1P AS=1P
MP6 COUT CN VDD VDD PMOS W=4U L=0.18U AD=2P AS=2P
CI1 I1 GND 2FF
CI3 I3 GND 3FF
CA A GND 4FF
CB B GND 4FF
CC C GND 2FF
CCN CN GND 4FF
CCOUT COUT GND 2FF
.ENDS
```

Physical Design

- Floorplan
- Standard cells
  - Place & route
- Datapaths
  - Slice planning
- Area estimation
MIPS floorplan

Does the design fit the chip area budgeted?
Estimates area of major units and defines their relative placement
Estimates wire lengths
Estimates wiring congestion

MIPS layout

- This design is pad limited (the I/O pads set the area of the chip)
- core-limited (the logic set the chip area)
Taxonomy of On-Chip structures

- Random logic
- Data paths
- Arrays
- Analog
- Input/output (I/O)

Synthesized MIPS Controller

**Fig. 1.03** MIPS controller layout
Hand-Crafted MIPS Datapath

FIG 1.66 MIPS datapath layout

Standard Cells

FIG 1.67 Simple standard-cell library. Color versions on inside front cover
Synthesized MIPS

Area Estimation

- Need area estimates to make floorplan
- Compare to another block you already designed
- Or estimate from transistor counts
- Budget room for large wiring tracks

<table>
<thead>
<tr>
<th>Table 1.10 Typical layout densities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element</td>
</tr>
<tr>
<td>random logic (2-level metal process)</td>
</tr>
<tr>
<td>datapath</td>
</tr>
<tr>
<td>SRAM</td>
</tr>
<tr>
<td>DRAM (in a DRAM process)</td>
</tr>
<tr>
<td>ROM</td>
</tr>
</tbody>
</table>

Some cell library vendor specify cell layout densities in Kgates/mm$^2$. 
**Design Verification**

- Fabrication is slow & expensive
  - MOSIS 0.6µm masks: $1000, 3 months
  - State of art masks (130nm): $1M, 1 month
- Debugging chips is very hard
  - Limited visibility into operation
- Prove design is right before building!
  - System simulation (C/C++)
  - Logic simulation (HDL testbench)
  - Circuit simulation / formal verification
  - Layout vs. schematic comparison
  - Design & electrical rule checks
- Verification is > 50% of effort on most chips!

**Fabrication**

- Tapeout final layout
  - Formats for mask descriptions: CIF (academia) and GDS II (industry)
- Fabrication
  - 6, 8, 12” wafers (bare wafer costs $1000-$5000)
  - Optimized for throughput, not latency (turnaround times up to 10 weeks !)
  - Cut into individual dice
- Fabs cost billions of dollars and become obsolete in a few years
  - Fabless semiconductor companies
  - Manufacturing Companies: TSMS, UMC, IBM
Packaging

- Bond gold wires from die I/O pads to package

FIG 1.71 Chip in a 40-pin dual-inline package

Testing

- Test that chip operates
  - Design errors
  - Manufacturing errors
- A single dust particle or wafer defect kills a die
  - Yields from 90% to < 10%
  - Depends on die size, maturity of process
  - Test each part before shipping to customer
Summary

- Many chip designers spend much of their time specifying circuits with HDL and seldom look at the actual transistor
- However:
  - Chip Design is not software engineering
  - It requires a fundamental understanding of circuit and physical design
- The best way to learn VLSI design is by doing it

Summary cont.

Now you know everything to start designing a simple chip!