ENEE 359a

Digital VLSI Design

Course Overview: Transistors to Systems

Prof. Bruce Jacob
blj@eng.umd.edu

Credit where credit is due:
Slides contain original artwork (© Jacob 2004) as well as material taken liberally from Irwin & Vijay’s CSE477 slides (PSU), Schmit & Strojwas’s 18-322 slides (CMU), Wolf’s slides for Modern VLSI Design, and/or Rabaey’s slides (UCB).
Why Digital?

Assume noise in your electronics ...
(lights flicker for no apparent reason, speakers pop when refrigerator turns on, cable station comes in fuzzy, etc.)

...and you want to eliminate/reduce problem

Analog: noise-induced errors propagate and accumulate (e.g., further noise?); hard to distinguish noisy signal from “true” signal
Why Digital?

How would *YOU* solve the problem?

- Problem arises because the entire range of output values is fair game as input
- Thus, it is hard to distinguish corrupted or noisy signal from “true” signal
Why Digital?

How would **YOU** solve the problem?

- ECC-protect all transmissions? (can’t, literally, but can do an analogy …)
- Use a smaller valid-data range so that small errors don’t matter
Why Digital?

How would YOU solve the problem?

- ECC does this numerically; here, we’re talking about interpreting voltage levels
- Relatively straight-forward to do this with various types of amplifiers
Why Digital?

Take this to the extreme:

BINARY (two values)
- Very robust in the face of noise
- Well-suited to BOOLEAN LOGIC

*** Dramatic oversimplification
Why Is This Robust?

These curves are said to be *regenerative*. Noise neither *propagates* nor *accumulates*.

This is exactly what digital logic does, e.g. chain of inverters:
How Is It Done? (devices)

Implementation: Transistors (switches)

Inverter Function:

Do I connect my output to “1” or “0”?

Each is effectively a signal repeater
How Is It Done? (devices)

MOS Transistors:

NMOS

PMOS

What's a “C” MOS?
How Is It Done? (devices)

MOS Transistor:

- p-doped semiconductor substrate
- PN Junction
- N-Doped Region [donor electrons]
How Is It Done? (devices)

MOS Transistor:

- p-doped semiconductor substrate
- P-Doped Region [acceptor holes]
- N-Doped Regions [donor electrons]
How Is It Done? (devices)

MOS Transistor:

- p-doped semiconductor substrate
- Insulator (gate oxide)
- VDD
- VSS
How Is It Done? (devices)

NMOS Transistor with gate:

- p-doped semiconductor substrate
- Insulator (gate oxide)
- Conductor
How Is It Done? (devices)

NMOS Transistor with bias voltages:

- p-doped semiconductor substrate
- Insulator (gate oxide)
- Conductor
How Is It Done? (devices)

NMOS Transistor with bias voltages:

p-doped semiconductor substrate
How Is It Done? (devices)

NMOS Transistor with bias voltages:

- p-doped semiconductor substrate
- Channel: n
- Source
- Drain
- Gate

Electron Flow:

- Source: 0 (Vss)
- Gate: 0
- Drain: V > 0
- Source: 0 (Vss)
- Gate: V > 0
- Drain: V > 0
How Is It Done? (devices)

PMOS Transistor with bias voltages:

- Drain Source
- Gate
- n-doped semiconductor substrate
- VDD

Electron Flow:
- Drain 0
- Gate VDD
- Source V > 0 (Vdd)

- Drain 0
- Gate V < VDD
- Source V > 0 (Vdd)

- Electron Flow
How Is It Done? (devices)

MOS Transistors:

NMOS

PMOS

CMOS Inverter = one of each
How Is It Done? (devices)

MOS Transistors:

NMOS

PMOS

CMOS Inverter = one of each
How ...? (manufacturing)

Wafer is a thin slice off a silicon log; Each wafer produces many identical chips
How ...? (manufacturing)

1. After oxidation and deposition of negative photoresist

2. Si-substrate
   Photoresist
   SiO₂
   Si-substrate

3. Stepper exposure

4. After development and etching of resist, chemical or plasma etch of SiO₂

5. After etching

6. Hardened resist
   SiO₂
   Si-substrate

7. Exposed resist

8. Final result after removal of resist

Chemical or plasma etch

Hardened resist

SiO₂

Si-substrate

SiO₂

Si-substrate

SiO₂

Si-substrate

SiO₂

Si-substrate
How ...? (manufacturing)

1. Create thin oxide in the “active” regions, thick elsewhere

2. Deposit polysilicon

3. Etch thin oxide from active region (poly acts as a mask for the diffusion)

4. Implant dopant
How ...? (manufacturing)

CMOS Inverter

![Diagram showing CMOS Inverter with N-Well and cut line]
How ...? (manufacturing)

- **N-regions** for source, drain
- **Gate (poly)**
- **P-regions and gate** for NMOS device
- **N-Well** ("n-well process")
- **P-type wafer**

CMOS Inverter

- **Cut line**
How ...? (manufacturing)

- **N-regions** for source, drain
- **Gate** (poly)
- **P-regions** and gate for PMOS device

**Diagram:**
- Input connected to NMOS
- Output connected to PMOS
- NMOS connected to GND
- PMOS connected to VDD
- N-Well
- Cut line
How …? (manufacturing)

Another view (note: wells/tubs not shown)
How Is It Done? (logic)

CMOS Inverter

VDD ("1")

\( s=0 \)
\( p \)
\( s=0 \)
\( n \)
GND ("0")

out = VDD ("1")

VDD ("1")

\( s=1 \)
\( p \)
\( s=1 \)
\( n \)
GND ("0")

\( \overline{s} \)
How Is It Done? (logic)

CMOS NAND-gate

VDD

s1
pmos

VDD

s2
pmos

output

VDD

nmos

s1

nmos

s2

GND

A

B

output

A

B

circuit

0 1
1 1 1
0 1

1 1 0
How Is It Done? (logic)

CMOS NOR-gate

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
How Is It Done? (logic)

CMOS AND–gate?

TEMPTING, BUT BAD IDEA (why?)
How Is It Done? (design)

Increasing clock rates, increasing densities, increasing design complexity, 1000s of I/O, decreasing design times …

Divide & Conquer:

- Modular, hierarchical design
- Well-defined interfaces
- Multiple levels of abstraction
- EXTENSIVE use of CAD tools …
- Back-end integration
How Is It Done? (design)

Big change in industry, late 1970’s:

   Mead & Conway introduced *design rules*

Previous to this, all IC work was by hand, and integration of components was at the chip level (CPU = tons of connected chips)

*Design rules enabled CAD tools*

Popular (and powerful) tools today:
- Cadence tools (behavioral-to-layout)
- Synopsys tools (synthesis)
- HSPICE (circuit simulation)
How ...? (tools: verilog)

The Many Faces of NOT (inverter)

Structural Verilog:

```verilog
wire s;
wire sbar;
NOT not1(s, sbar);
```

Behavioral Verilog:

```verilog
wire s, sbar;
sbar = ~s;
```
How ...? (tools: verilog)

The Many Faces of NAND

**Structural Verilog:**

```verilog
wire a, b;
wire out;
NAND nand1(a, b, out);
```

**Behavioral Verilog:**

```verilog
wire a, b, out;
out = ~(a & b);
```
**How …? (tools: synopsys)**

**Logic synthesis: Behavioral -> Structural**

```plaintext
wire a, b, out;
out = ~(a & b);
```

**Logic Synthesis**

```plaintext
wire a, b;
wire out;
NAND nand1(a, b, out);
```

**Library Instantiation**
How ...? (tools: synopsys)

Logic synthesis: *Behavioral* -> *Structural*

```vhdl
wire a, b, c, out;
out = ~((a & b) | c);
```

**Logic Synthesis**

```vhdl
wire a, b, c, out;
wire t1, t2;
AND and1(a, b, t1);
OR or1(c, t1, t2);
NOT not1(t2, out);
out = [ab+C];
```

**Library Instantiation**

**LAYOUT**
How ...? (tools: synopsys)

The tool will provide design options ...

```plaintext
wire a, b, out;
out = a + b;
```

Four Adders:

- Ripple
- Brent-Kung
- Carry-Lookahead
- Fast Carry-Lookahead
How ...? (tools: HSPICE)

Very accurate simulation of circuits
How …? (tools: cadence)

- Full custom, die photo
- Semi-custom, die photo
- Fully synthesized, layout
- Mostly custom, layout
How Is It Done? (design flow)

Begin with a Behavioral Description:

```verilog
module testcounter(clk2, rst_l, out_w);

input clk2, rst_l;
output[7:0] out_w;

reg [7:0] src1, out;
wire [7:0] out_w = out;

always @(posedge clk2)
begin
    if(!rst_l)
        begin
            src1 <= 1'd0;
            out <= 1'd1;
        end
    else
        begin
            src1 <= out_w;
            out <= src1 + out_w;
        end
end
endmodule
```
How Is It Done? (design flow)

Logic synthesis produces gate-level netlist:
How Is It Done? (design flow)

Instantiation of physical design libraries & place+route yields Physical Layout:

(two views shown: one with black boxes for high-level structures like flip-flops/adders/MUXes, the other showing all transistors and wires)
How Is It Done? (design flow)

Send to fabrication facility, receive Chip:

testcounter
How Is It Done? (system)

Simplest possible system-building scheme:

*Sequential—Do One Thing At a Time*

Advantages:
- Simple, predictable, easy to debug

Disadvantages:
- Slow, wastes hardware (f & h blocks idle while f computes, f & h blocks idle while g computes, etc.)
How Is It Done? (system)

Called a **PIPELINE**: extremely common
- Hardware used as fully as possible
- Throughput increases n-fold (n blocks)

Question: How to control this?
How Is It Done? (system)

**VERSUS**

Control: Think *traffic in a city* (stop lights)

Clock synchronizes movement of data
What Are The Gotcha’s?

“High-speed digital design, in contrast to digital design at low speeds, emphasizes the behavior of passive circuit elements. These passive elements may include the wires, circuit boards, and integrated-circuit packages that make up a digital product. At low speeds, passive circuit elements are just part of a product’s packaging. At higher speeds the directly affect electrical performance.”

— opening par. of Johnson & Graham, High-Speed Digital Design

The bottom line:

At high speeds, digital systems (which diverged from analog: simpler, remember?) start to behave just like analog systems.
What Are The *Gotcha’s*?

**HIGH-SPEED DESIGN exposes problems:**

- Even small amounts of **NOISE**
- Even small amounts of **DELAY**
- Even small amounts of **CURRENT**

... can cause a circuit to misbehave.

- At high speeds, signal levels are small (small noise levels become significant)
- At high speeds, event timing is tight (small errors in time become significant)
- At high speeds, current changes quickly (even if $di$ is small, $Ldi/dt$ can be large)
**Gotcha’s? (noise)**

**Typical noise sources:**

- **Inductive coupling**
- **Capacitive coupling**
- **Groundplane noise**
- **Implicit/explicit circuits** (current return) make great radiators and antennae
Gotcha’s? (delay)

Though we like to imagine that it is, signal propagation is not instantaneous
Gotcha’s? (delay)

Argument for source-synchronous clocking:

If CLK is used to sample data*, this adds 0.14ns to uncertainty

("def’n: tell A when to look at data bus)
Gotcha’s? (current)

Ideal Scenario vs. Reality (ground bounce):

Current flow changes direction when input (thus output) values change

Magnitude of current change is $di$

The time to switch directions is $dt$

The voltage-drop induced on this wire at time of switching is $L \frac{di}{dt}$
Gotcha’s? (current)

Simultaneous Switching Noise:

- FCRAM side
  - VDDQ(Pad)
  - VSSQ(Pad)
  - DQS(Pin)
  - DQ0-15(Pin)
- Controller side
  - DQS(Pin)
  - DQ0-15(Pin)
  - skew=158psec
  - skew=102psec
What Are Some Solutions?

IN GENERAL:

Prevent the Problem

- Design as if the problem doesn’t exist (continue using same old techniques as you did with low-speed designs)
- Use mechanism to counteract problem

Design Around the Problem

- Re-think the way you design systems: assume the problem happens; ensure that it doesn’t affect design (design makes no assumptions re: problem)
- Don’t need to counteract problem
What Are Some Solutions?

One solution to delay problem: 

*heroic routing*

[Diagram of circuit with labeled paths and signal connections]
What Are Some Solutions?

Another solution to the delay problem

Sliding Time

$0^{th}$ \quad $\cdots$ \quad $N^{th}$

Clk SRC

H Tree

$0^{th}$ \quad $\cdots$ \quad $N^{th}$

Clk SRC
What Are Some Solutions?
(clock trees work, but may be better sol’ns?)
What Are Some Solutions?

One solution to noise problem (shielding):
What Are Some Solutions?

Another solution to the noise problem:

- Single Ended Transmission Line
- Differential Pair Transmission Line
What Are Some Solutions?

(differential signaling works quite well)
What Are Some Solutions?

Another type of shielding:

Couples extremely well with previous sol’n (less effective if used for single-ended transmission)
What Are Some Solutions?

IN GENERAL:

Prevent the Problem

- Requires no change in design practice; no learning curve, no increase in NRE design costs
- The mechanism might be expensive or skittish, or both

Design Around the Problem

- Requires engineering creativity, willingness to be unconventional
- First attempt might not succeed
- Cost/reliability *might* be better (long run)
**Historical**

Digital design has been an enormously successful paradigm, driven largely by the ability to use CAD tools to verify designs.

Result: *exponential increases* in design complexity that were predicted in 1965* and have continued for *FOUR DECADES*

---

*Moore’s Law.* 1965: transistors on chip doubles every year. Revised in 1975: number of transistors doubles every two years.
Historical

Intel 4004

1971
1000 transistors
800 KHz operation
Historical

Intel 8086

1978
29K trans.
10 MHz
Historical

Intel 80286

1982
134K trans.
12 MHz
Historical

Intel 80386

1985/89
275K trans.
16/33 MHz
Historical

Intel 80486

1992
1.6M trans.
100 MHz
Historical

Intel Pentium

1993/99
3.1/4.5M trans.
60/300 MHz
Historical

Intel Pentium Pro
Historical

Intel Pentium IV

2000/04
42/178M trans.
1.4/3.6 GHz
Big Picture

Digital vs. Analog:
- simple vs. complex
- robust vs. fragile

BUT: *This is Only True at Low Speeds*
(And Nothing is Low-Speed Any More)

Analogy of Architect and the Carpenter:
- Architect does a better job if he knows carpentry
- Digital systems design can’t be done without using analog concepts