

(Lec 0) Introduction: 18-760 Fall 2001

▼ VLSI CAD: Logic to Layout

- ▶ Faculty: Rob Rutenbar x8-3334 HH3015 rutenbar@ece.cmu.edu
- ▶ TA: *still looking...*
- ▶ Secretary: Lyz Knight x85087 HH3107 lyz@ece.cmu.edu
- ▶ URL: <http://www.ece.cmu.edu/~ee760>

▼ Course materials

- ▶ Book
 - ▶ *Synthesis and Optimization of Digital Circuits, Giovanni DeMicheli*
- ▶ Notes
 - ▶ I teach on color foils (like *this one...*)
 - ▶ You get hard copy at beginning of each class
 - ▶ Old copies of stuff shelf on 3rd floor of HH, and pdf's on web

Obligatory Copyright Notice

▼ I put this stuff out on the web, so every talk has this:

© Rob A. Rutenbar, 2001
All rights reserved.

You may not make copies of this material in any form without my express permission.

Handouts

Physical

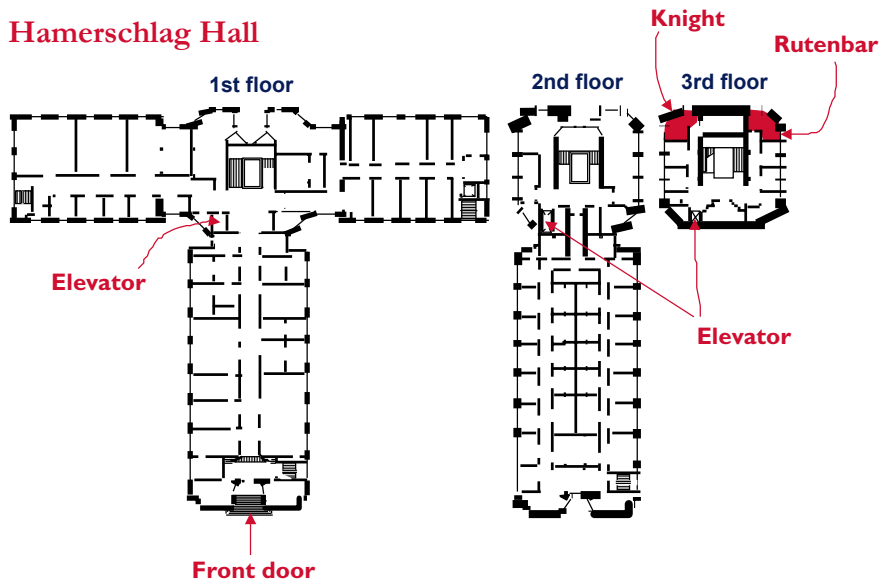
- ▶ Lecture 00 -- Introduction
- ▶ Lecture 01 -- Advanced Boolean Algebra
- ▶ Syllabus

Electronic

- ▶ Go look on the web site to get the 2-page course **syllabus** as a pdf

Aids for the Navigationally Impaired

Hamerschlag Hall



Pre-Requisites

▼ Computer science knowledge

- ▶ Programming and data structures (at CMU, 15-211, maybe also 15-213)
- ▶ Basic complexity ideas, eg, why $O(n^2)$ is worse than $O(n \log n)$
- ▶ Ability to write a medium size program -- 1000-2500 lines of C or C++ code -- without any hand holding from me, in about 3 weeks

▼ Computer engineering knowledge

- ▶ Basic digital design (gates, flip flops, Boolean algebra, Kmaps)
- ▶ Combinational and sequential design (finite state machines)

▼ Discrete math

- ▶ Basic sets, functions, careful notation
- ▶ Exposure to graph theory is nice but not essential (15-211 is fine)

Grading

▼ No real exams

▼ Homeworks (40%)

- ▶ About 6 in all, about every other week (with a few breaks)
- ▶ Mechanical work-it-through problems, small proofs, algorithm manipulations
- ▶ Small programs (300-400 lines of code) to try things out

▼ Projects (45%)

- ▶ 1 smaller one (5%) and 2 larger ones (20%, 20%)
- ▶ Bigger, ~2500 lines of code typically, done in groups of 2

▼ Papers (15%)

- ▶ 3 papers you read and write about
- ▶ Short reviews (4-pages or 10-Powerpoint slides) of a paper from the literature, or compare and contrast of 2 papers from literature

Undergrad versus Grad Students

▼ Undergrads

- ▶ Expectation is for you to keep up, absorb the “big ideas”

▼ Grads

- ▶ Expectation is for you to absorb it *all*

▼ Note

- ▶ Diverse class this year, as usual
- ▶ Some maybe took 18-360 (which uses versions of my 760 notes!)
- ▶ Some folks complete rookies
- ▶ My problem to keep it all *interesting* for everybody...

So What's the Course All About...?

▼ CAD for semi-custom ASICs

- ▶ ASIC = application-specific integrated circuit
- ▶ Semi-custom = try to design reusing some already designed parts
- ▶ CAD = flow through a sequence of design steps and software tools



Buzzword Acronym Lexicon

▼ Semi-custom ASIC

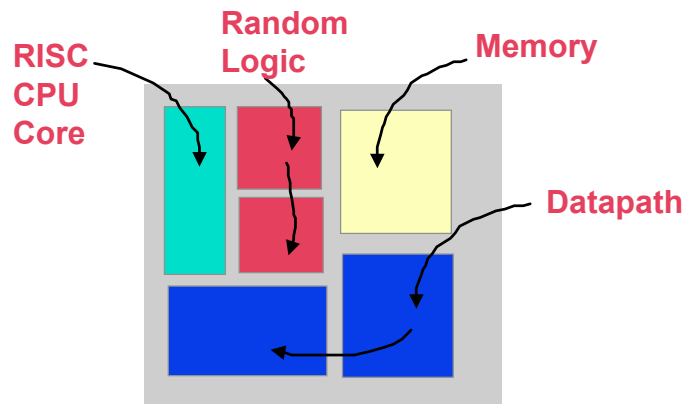
- ▶ *Application-specific IC* - when you design a chip for a specific task using mostly semi-custom techniques
- ▶ Don't expect to make a zillion of them, so can't afford full custom; **OR** you need a chip really quick, so can't afford full custom
- ▶ Not as dense (transistors / area) or as fast (MHz) as full custom

▼ Full custom IC

- ▶ Well, really a misnomer, since almost nothing is absolutely custom, completely done by hand
- ▶ Almost every custom chip has big chunks of semi-custom function since there are good tools for this stuff
- ▶ Examples: your favorite microprocessor

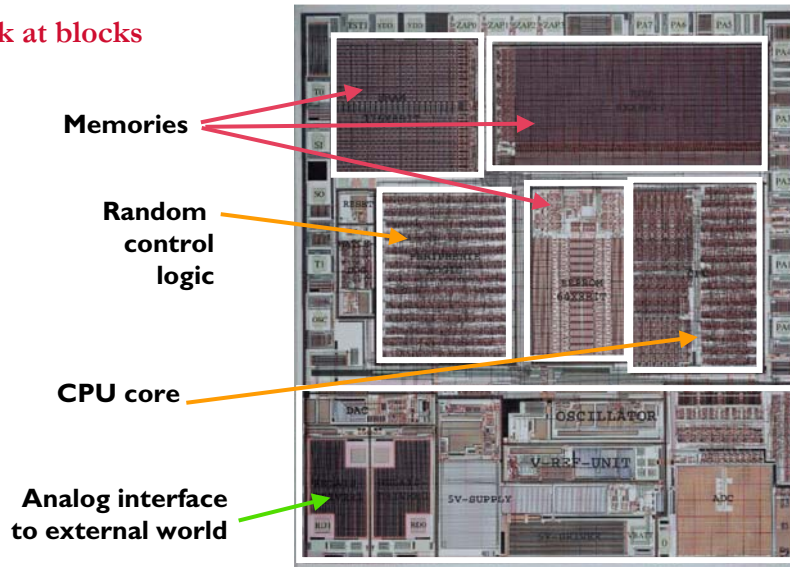
Example Modern System-on-a-Chip IC

▼ Lots of big, separate chunks



Real Example: Called “System on Chip” or SoC

▼ Look at blocks



© R. Rutenbar 2001

Fall 18-760 Page 13

What are We Focusing On?

▼ Gates as the central components of technology

- ▶ This is where most of the interesting algorithms are
- ▶ Also, prevents us from having to go do any serious “electricity” stuff

▼ What are gates good for?

- ▶ Blobs of random logic
- ▶ Techniques are great for controllers, so-called glue logic, bus interfaces, finite state machines, etc

▼ What are gates less good for?

- ▶ Datapaths (arithmetic) where you can usually do better by designing at the transistor level
- ▶ But—people do *really* use gates for lots of real datapaths

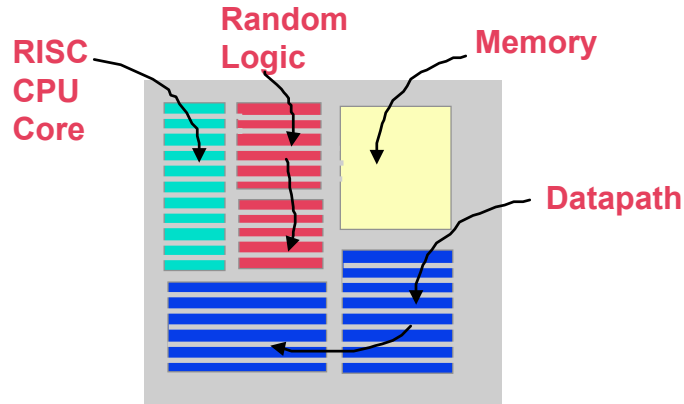
© R. Rutenbar 2001

Fall 18-760 Page 14

Semi-Custom ASICs

▼ Mostly made out of so-called “standard cells”

- ▶ Standard cell = 1 gate (might be a complex gate, though)
- ▶ Usually arranged in rows on surface of the IC



© R. Rutenbar 2001

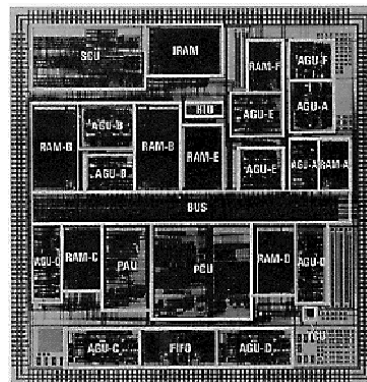
Fall 18-760 Page 15

Example Semi-Custom ASIC

▼ Possible to do *everything* as standard cells

- ▶ Less dense, less performance than custom
- ▶ But certainly possible. Indeed, maybe desirable if your concern is designing something really quickly to get a product out
- ▶ Can do really complicated stuff, fast, this way
- ▶ Example: most PC 3D graphics chips done like this

Example Inoue et al,
Dec 1993 JSSC,
300MHz 16 bit BiMOS
Video Signal Processor



© R. Rutenbar 2001

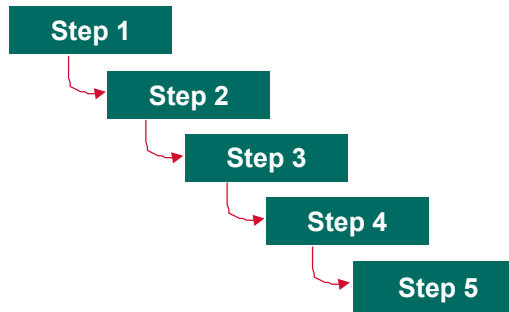
Fall 18-760 Page 16

CAD Flow

▼ So how do people attack designs like this?

▼ Big idea: *Levels of abstraction*

- ▶ Break the problem down into a lot of smaller steps
- ▶ Each step renders the design a little less abstract, a little more real
- ▶ Synthesis tools go forward in design process: make new stuff
- ▶ Verification steps look backward in process: check that stuff worked

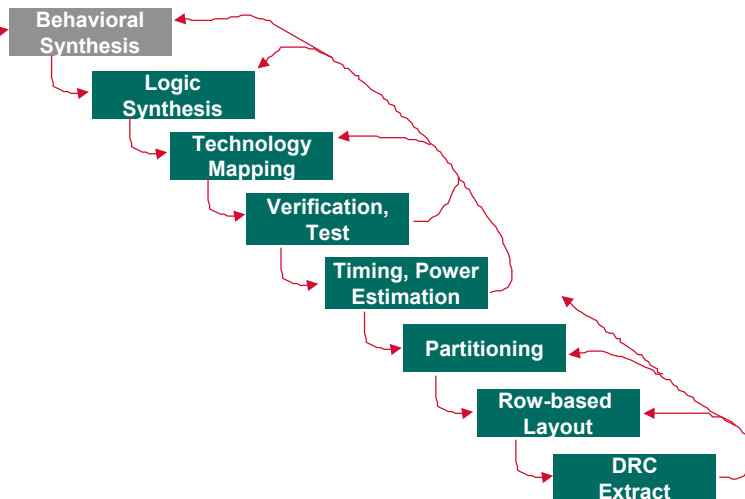


© R. Rutenbar 2001

Fall 18-760 Page 17

ASIC CAD Tool Flow

High-level description



© R. Rutenbar 2001

Fall 18-760 Page 18

High-Level (Behavioral) Synthesis

What goes in?

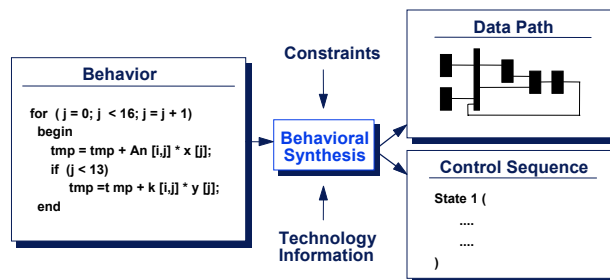
- ▶ High-level description of desired system function, usually as a program in a hardware description language like Verilog or VHDL

What comes out?

- ▶ Register transfer level structure: FSMs, logic, ALUs, memory, busses

Our coverage?

- ▶ Zero. Sorry, no time.



© R. Rutenbar 2001

Fall 18-760 Page 19

Logic Synthesis

What goes in?

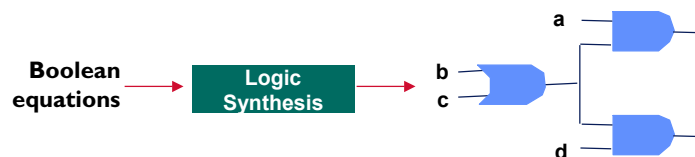
- ▶ Boolean equations, state diagrams, etc

What comes out?

- ▶ Gates and connections - called a *netlist*, a *structural design*

Our coverage

- ▶ About 6 lecs + 2 background



© R. Rutenbar 2001

Fall 18-760 Page 20

Technology Mapping

What goes in?

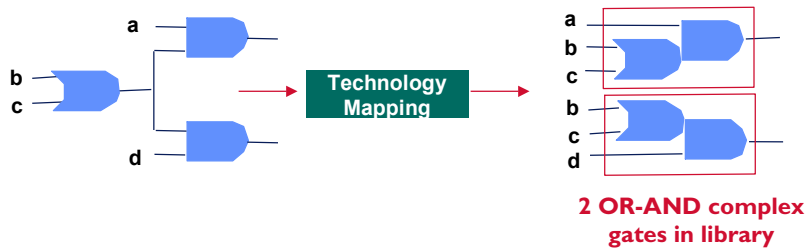
- ▶ Technology-independent gate-level design (so-called *uncommitted design*)

What comes out?

- ▶ Gate level design that uses *your* gate technology library

Our coverage

- ▶ About 1 lecs



© R. Rutenbar 2001

Fall 18-760 Page 21

Formal Verification

What goes in?

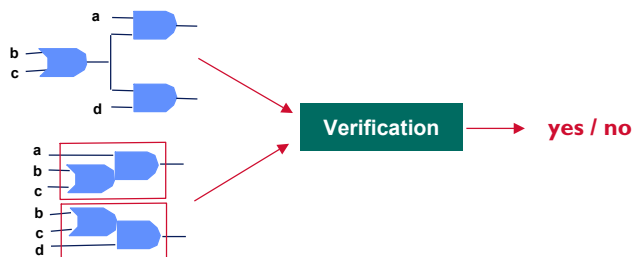
- ▶ A specification for a design (eg, Boolean eqns) and an implementation of the design (eg, gates)

What comes out?

- ▶ A decision yes/no: is specification == implementation, same function?
- ▶ Note, “formal” verification means it’s *proved* yes/no, unlike simulation

Our coverage

- ▶ About 5 lecs, + 2 background



© R. Rutenbar 2001

Fall 18-760 Page 22

Test (Automatic Test Pattern Generation, ATPG)

What goes in?

- ▶ A gate level design
- ▶ List of possible faults in the circuit you wish to be able to discover

What comes out?

- ▶ Patterns of inputs that will make the circuit produce a wrong answer if any of these faults are present in the manufactured part

Our coverage

- ▶ Again, none -- go look at 18-765



© R. Rutenbar 2001

Fall 18-760 Page 23

Timing Estimation

What goes in?

- ▶ A gate level design, with timing info about gates and wires

What comes out?

- ▶ Delay estimate -- how long to go through this logic network?

Our coverage

- ▶ About 2 lecs (logical timing)



© R. Rutenbar 2001

Fall 18-760 Page 24

Row-Based Semi-Custom Layout

What goes in?

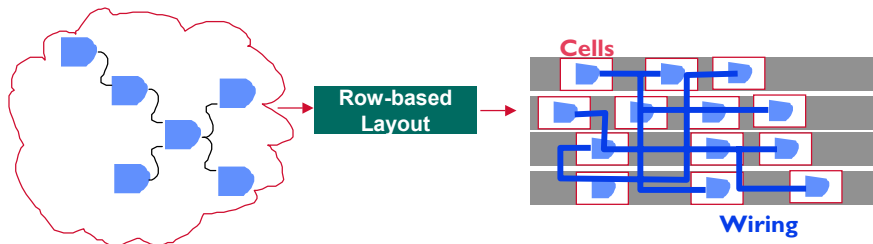
- ▶ A gate level design, gates+wires

What comes out?

- ▶ A placement of cells into rows, with wiring over the top of the cells

Our coverage

- ▶ About 7 lecs



© R. Rutenbar 2001

Fall 18-760 Page 25

Electrical Timing Issues

(OK, I lied when I said “no electricity”...)

What goes in?

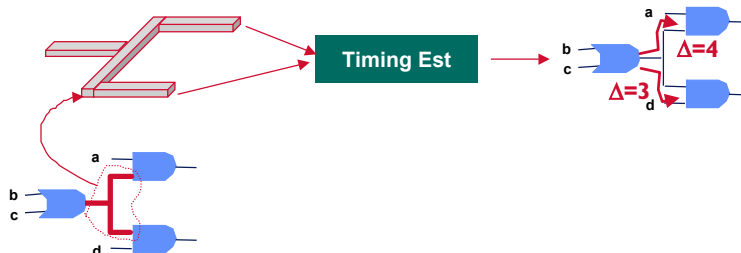
- ▶ Geometric information about actual wiring patterns

What comes out?

- ▶ Delay estimate -- how long to go through this *electrical* network?

Our coverage

- ▶ 1 lec



© R. Rutenbar 2001

Fall 18-760 Page 26

Mask Geometry: Representing & Checking

What goes in?

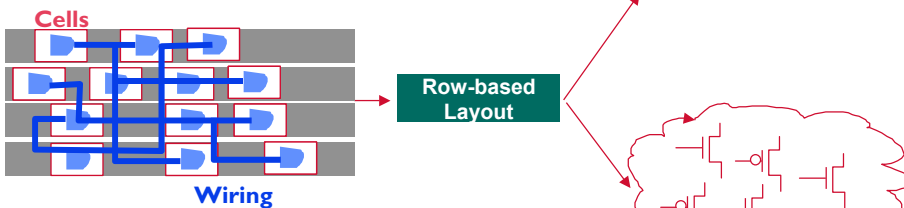
- ▶ A mask, ie, geometry of layout of an IC

What comes out?

- ▶ Location of violations of rules in masks, and extracted netlist of what devices the mask actually implements

Our coverage

- ▶ About 3 lecs



© R. Rutenbar 2001

Fall 18-760 Page 27

18-760 Topical Summary

760 is about the *flow* of tools used to do modern semi-custom ASICs

- ▶ Cover core algorithms in detail
- ▶ Both synthesis and verification
- ▶ Emphasis is Boolean stuff, timing stuff, rectangles stuff
- ▶ Only a very little EE-type electricity stuff

What else?

- ▶ Context and “ambience”: what’s cool, what’s easy, what’s hard, etc
- ▶ Where the complete solutions are, where we have to use heuristics
- ▶ You should acquire the ability to read the CAD literature from 760

© R. Rutenbar 2001

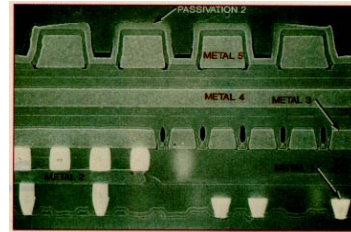
Fall 18-760 Page 28

18-760 Topical Summary

Why is this cool?

Increasingly, the problem is methodology -- the tool flow

- ▶ Individual algorithms work OK alone...
- ▶ ..but together the overall result of the "flow" doesn't work



5-layer metal cross section from IBM PowerPC

Example: Nanometer CMOS scaling

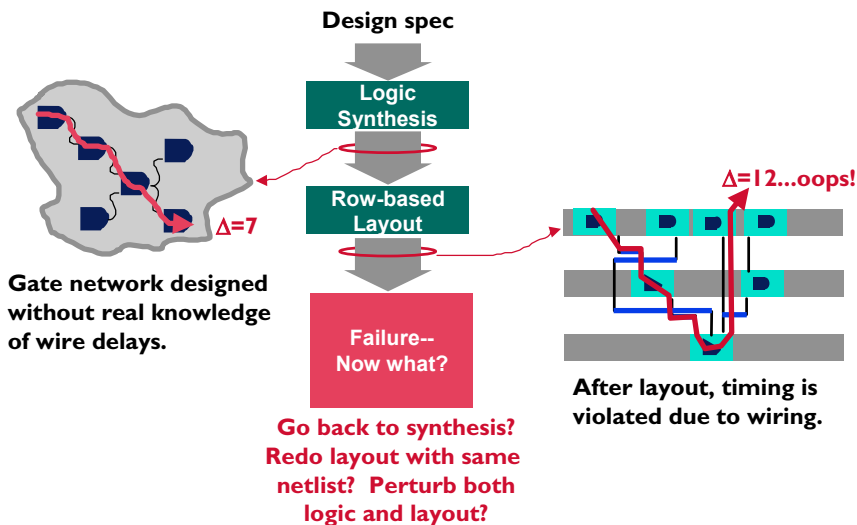
- ▶ Individual devices and wires are very small now, $\ll 1 \mu\text{m}$
- ▶ Next-gen CMOS technology $\sim 0.1 \mu\text{m}$
 ~ 200 atoms wide wires & FET-gates
- ▶ Nasty physics dominates: much of the delay is from the wires and not the gates, nearby devices affect each other, etc

© R. Rutenbar 2001

Fall 18-760 Page 29

Nanometer Technology Example

Convergence problems between synthesis & layout



© R. Rutenbar 2001

Fall 18-760 Page 30

Now What?

▼ On to Boolean algebra...

	M	T	W	Th	F	
Aug	27	28	29	30	31	1
Sep	3	4	5	6	7	2
	10	11	12	13	14	3
	17	18	19	20	21	4
	24	25	26	27	28	5
Oct	1	2	3	4	5	6
	8	9	10	11	12	7
	15	16	17	18	19	8
	22	23	24	25	26	9
	29	30	31	1	2	10
Nov	5	6	7	8	9	11
	12	13	14	15	16	12
Thnxgive	19	20	21	22	23	13
	26	27	28	29	30	14
Dec	3	4	5	6	7	15
	10	11	12	13	14	16

Introduction

- Advanced Boolean algebra
- JAVA Review
- Formal verification
- 2-Level logic synthesis
- Multi-level logic synthesis
- Technology mapping
- Placement
- Routing
- Static timing analysis
- Electrical timing analysis
- Geometric data structs & apps