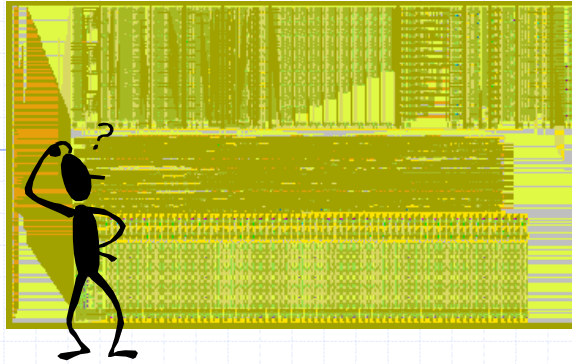


## 6.375 Complex Digital System Spring 2006



Lecturer: Arvind  
TAs: Chris Batten & Mike Pellauer  
Assistant: Sally Lee

February 8, 2006

<http://csg.csail.mit.edu/6.375/>

L01-1

## Do we need more chips (ASICs)?

ASIC=Application Specific IC

Some exciting possibilities  
based on research @ CSAIL

February 8, 2006

<http://csg.csail.mit.edu/6.375/>

L01-2

## Content distribution and customer service



Interactive, lifelike avatars as actors, news anchors, and customer service representatives

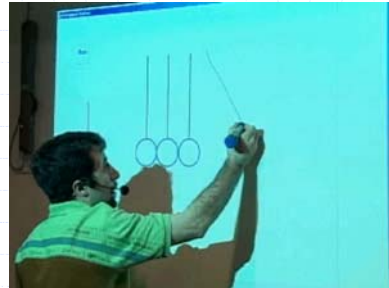
Source: Computer Science and Artificial Intelligence Laboratory at MIT (CSAIL)

February 8, 2006

<http://csg.csail.mit.edu/6.375/>

L01-3

## Ubiquitous, behind-the-scenes computing



Computer interfaces woven tightly into the environment

Source: Computer Science and Artificial Intelligence Laboratory at MIT (CSAIL)

February 8, 2006

<http://csg.csail.mit.edu/6.375/>

L01-4

# What's required?

ICs with dramatically higher performance, optimized for applications



and at a

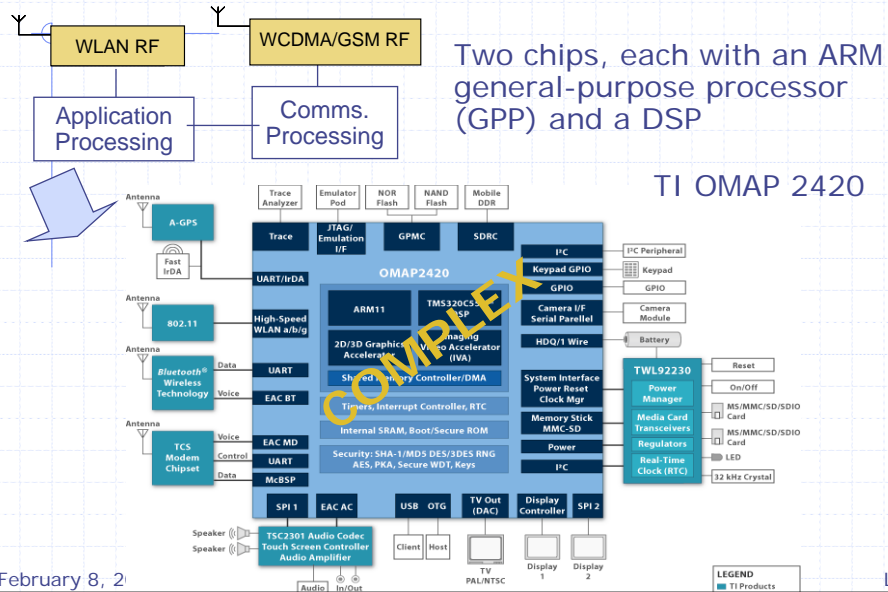
- ◆ size and power to deliver mobility;
- ◆ cost to address mass consumer markets

February 8, 2006

<http://csg.csail.mit.edu/6.375/>

L01-5

# Current Cellphone Architecture



February 8, 2006

L01-6

# Chip design has become too risky a business

- ◆ Ever increasing size and complexity
  - Microprocessors: 100M gates  $\Rightarrow$  1000M gates
  - ASICs: 5M to 10M gates  $\Rightarrow$  50M to 100M gates
  
- ◆ Ever increasing costs and design team sizes
  - > \$10M for a 10M gate ASIC
  - > \$1M per re-spin in case of an error (does not include the redesign costs, which can be substantial)
  
- ◆ 18 months to design but *only an eight-month selling opportunity in the market*
  - $\Rightarrow$ 
    - Fewer new chip-starts every year
    - Looking for alternatives, e.g., FPGA's

February 8, 2006

<http://csg.csail.mit.edu/6.375/>

L01-7

# Designer's Dilemma

## ASIC Complexity

- 2000: 1M+ logic gates
- 2005: 10M+ logic gates
- 2010: 100M+ logic gates

## Constants

- 10-30 person design team size
- 18 month design schedule
- *Design flow -- unchanged for 10+ years!*

## Designer must take shortcuts

- Conservative design
- No time for exploration
- Educated guess & code
- Gates are free mentality

PM Pipeline example

Which is best?

Pipeline	Area (mm <sup>2</sup> )	Gate Util (%)	Memory Util (%)
Static	8.898	3.60	63.5
Linear	15.718	4.70	99.9
Circular	8.1	3.67	99.9
Static (2)			63.5

[ICCAD'04]

What happens when a designer must implement a 1M gate block?

Sub-optimal implementations!

Alternatives?

February 8, 2006

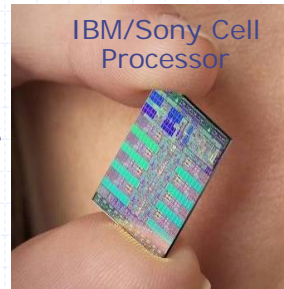
<http://csg.csail.mit.edu/6.375/>

L01-8

## One prevailing viewpoint: A sea of general purpose processors

### Advantages

- Easier to scale hardware design as complexity is contained within processors
- Easy to program and debug complex applications



### Disadvantages (as compared to an ASIC)

- Power ~100-1000X worse
- Performance up to ~100X worse
- Area up to ~10-100X greater

*Do we really know how to program these?*

February 8, 2006

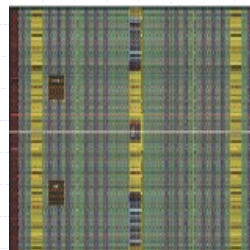
<http://csg.csail.mit.edu/6.375/>

L01-9

## Another popular "platform" vision: Field-Programmable Gate Arrays

### Advantages

- Dramatically reduce the cost of errors
- Remove the reticle costs from each design



### Disadvantages (as compared to an ASIC)

*[Kuon & Rose, FPGA2006]*

- Switching power around ~12X worse
- Performance up 3-4X worse
- Area 20-40X greater

*Still requires tremendous design effort at RTL level*

February 8, 2006

<http://csg.csail.mit.edu/6.375/>

L01-10

## Future could be different if we became 10X more productive in design

- ◆ This course is about new ways expressing behavior to reduce design complexity

- ◆ Decentralize complexity: *Rule-based specifications (Guarded Atomic Actions)*
  - Let us think about one rule at a time
- ◆ Formalize composition: *Modules with guarded interfaces*
  - Automatically manage and ensure the correctness of connectivity, i.e., correct-by-construction methodology
  - Retain resilience to changes in design or layout, e.g. compute latency  $\Delta$ 's
  - Promote regularity of layout at macro level

February 8, 2006

<http://csg.csail.mit.edu/6.375/>

L01-11

Let's take a look at the current CMOS technology...

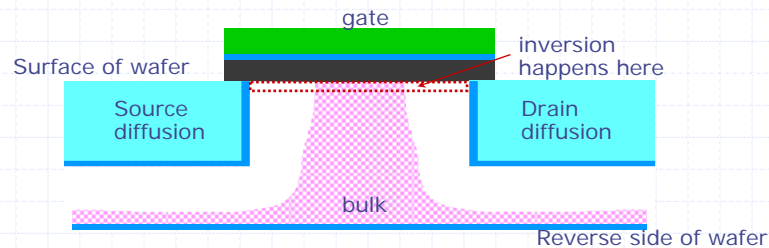
February 8, 2006

<http://csg.csail.mit.edu/6.375/>

L01-12

# FET = Field-Effect Transistor

A four terminal device (gate, source, drain, bulk)



Inversion: A vertical field creates a channel between the source and drain.

Conduction: If a channel exists, a horizontal field causes a drift current from the drain to the source.

February 8, 2006

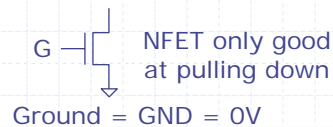
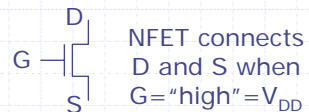
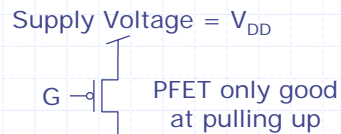
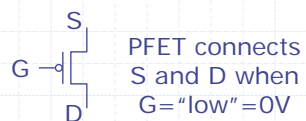
<http://csg.csail.mit.edu/6.375/>

L01-13

# Simplified FET Model

Binary logic values represented by voltages:

"High" = Supply Voltage, "Low" = Ground Voltage

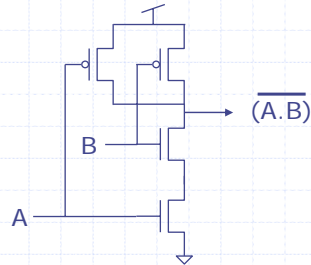
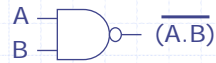


February 8, 2006

<http://csg.csail.mit.edu/6.375/>

L01-14

# NAND Gate



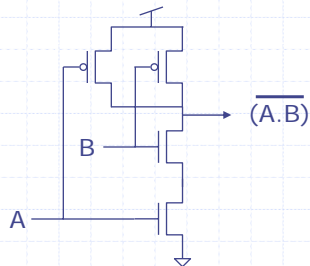
- When both A and B are high, output is low
- When either A or B is low, output is high

February 8, 2006

<http://csg.csail.mit.edu/6.375/>

L01-15

# NAND Gate Layout



Poly wire connects PMOS & NMOS gates

Metal 1-Diffusion Contact

Parallel PMOS Transistors  
 $V_{DD}$   
P-Diffusion (in N-well)

N-Diffusion  
Series NMOS Transistors

Output on Metal-1

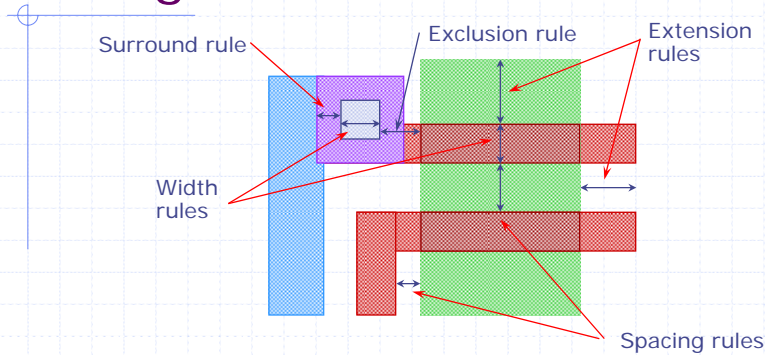
February 8, 2006

<http://csg.csail.mit.edu/6.375/>

L01-16



# Design Rules



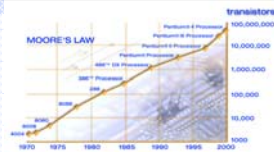
- ◆ An abstraction of the fabrication process that specify various geometric constraints on how different masks can be drawn
- ◆ Design rules can be absolute measurements (e.g. in nm) or scaled to an abstract unit, the *lambda*. The value of *lambda* depends on the manufacturing process finally used.

February 8, 2006

<http://csg.csail.mit.edu/6.375/>

L01-17

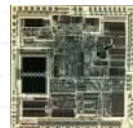
# Exponential growth: Moore's Law



Intel 8080A, 1974  
3Mhz, 6K transistors, 6u



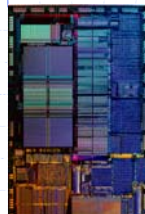
Intel 8086, 1978, 33mm<sup>2</sup>  
10Mhz, 29K transistors, 3u



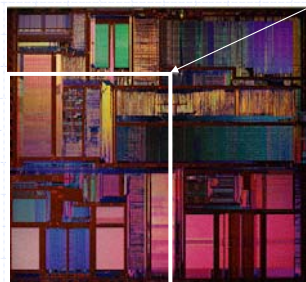
Intel 80286, 1982, 47mm<sup>2</sup>  
12.5Mhz, 134K transistors, 1.5u



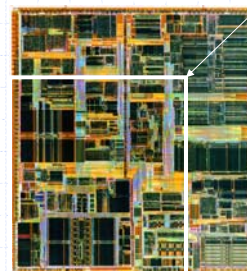
Intel 386DX, 1985, 43mm<sup>2</sup>  
33Mhz, 275K transistors, 1u



Intel 486, 1989, 81mm<sup>2</sup>  
50Mhz, 1.2M transistors, .8u



Intel Pentium, 1993/1994/1996, 295/147/90mm<sup>2</sup>  
66Mhz, 3.1M transistors, .8u/.6u/.35u



Intel Pentium II, 1997, 203mm<sup>2</sup>/104mm<sup>2</sup>  
300/333Mhz, 7.5M transistors, .35u/.25u

Shown with approximate relative sizes

[http://www.intel.com/intel/intelis/museum/exhibit/hist\\_micro/hof/hof\\_main.htm](http://www.intel.com/intel/intelis/museum/exhibit/hist_micro/hof/hof_main.htm)

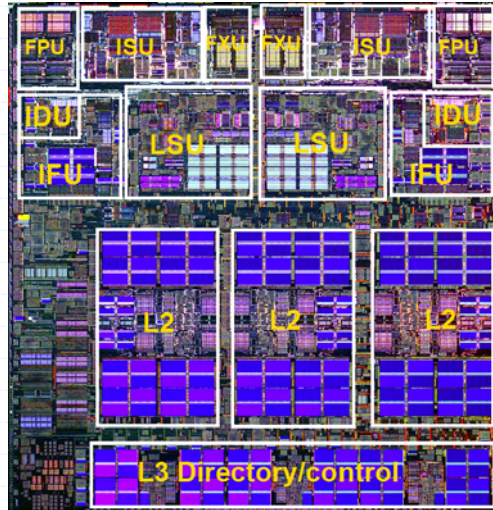
February 8, 2006

<http://csg.csail.mit.edu/6.375/>

L01-18

# IBM Power 5

- ◆ 130nm SOI CMOS with Cu
- ◆ 389mm<sup>2</sup>
- ◆ 2GHz
- ◆ 276 million transistors
- ◆ Dual processor cores
- ◆ 1.92 MB on-chip L2 cache
- ◆ 8-way superscalar
- ◆ 2-way simultaneous multithreading

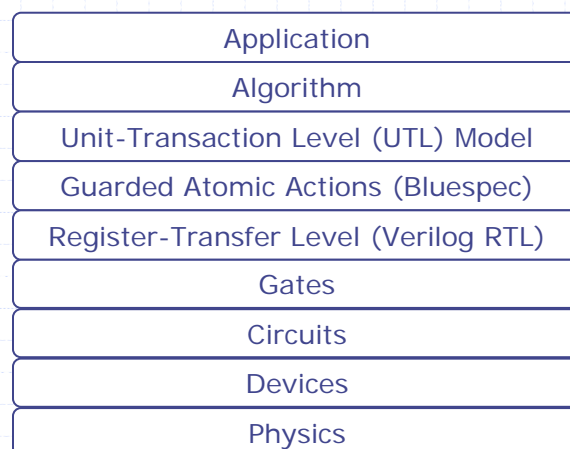


February 8, 2006

<http://csg.csail.mit.edu/6.375/>

L01-19

# Hardware Design Abstraction Levels



February 8, 2006

<http://csg.csail.mit.edu/6.375/>

L01-20

Tools play a crucial role  
in our ability to design  
economically

February 8, 2006

<http://csg.csail.mit.edu/6.375/>

L01-21

## ASIC Design Styles

- ◆ Full-Custom (every transistor hand-drawn)
  - Best performance: as used by Intel  $\mu$ Ps
- ◆ Semi-Custom (Some custom + some cell-based design)
  - Reduced design effort: AMD  $\mu$ Ps plus recent Intel  $\mu$ Ps
- ◆ Cell-Based ASICs (Only use cells in standard library)
  - This is what we'll use in 6.375
- ◆ Mask Programmed Gate Arrays
  - Popular for medium-volume, moderate performance applications
- ◆ Field Programmable Gate Arrays
  - Popular for low-volume, low-moderate performance applications
- ◆ Comparing styles:
  - how much freedom to develop own circuits?
  - how many design-specific mask layers per ASIC?

February 8, 2006

<http://csg.csail.mit.edu/6.375/>

L01-22

## Custom and Semi-Custom

- ◆ Usually, in-house design team develops own libraries of cells for commonly used components:
  - memories
  - register files
  - datapath cells
  - random logic cells
  - repeaters
  - clock buffers
  - I/O pads
- ◆ In extreme cases, every transistor instance can be individually sized (\$\$\$\$)
  - approach used in Alpha microprocessor development
- ◆ The trend is towards greater use of semi-custom design style
  - use a few great circuit designers to create cells
  - redirect most effort at microarchitecture and cell placement to keep wires short

February 8, 2006

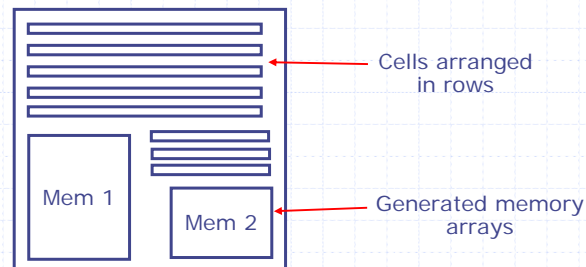
<http://csg.csail.mit.edu/6.375/>

L01-23

## Standard Cell ASICs

aka Cell-Based ICs (CBICs)

- ◆ Fixed library of cells + memory generators
- ◆ Cells can be synthesized from HDL, or entered in schematics
- ◆ Cells placed and routed automatically
- ◆ Requires complete set of custom masks for each design
- ◆ Currently most popular hard-wired ASIC type (6.375 will use this)

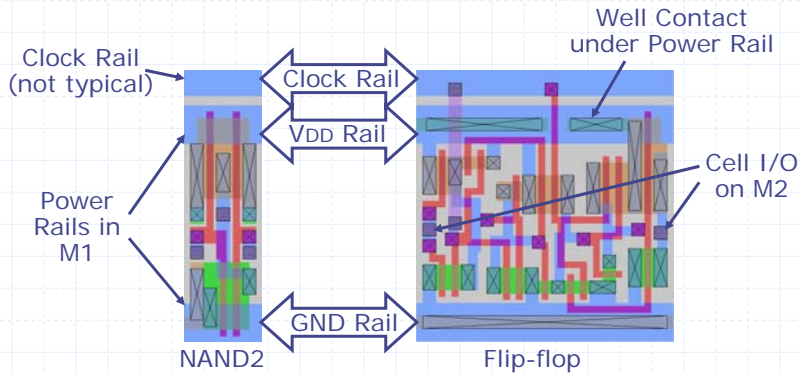


February 8, 2006

<http://csg.csail.mit.edu/6.375/>

L01-24

# Standard Cell Design



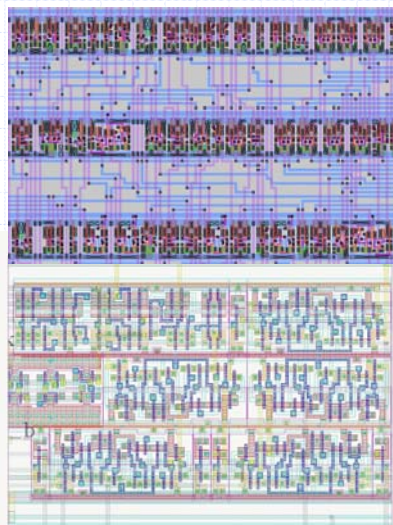
- ◆ Cells have standard height but vary in width
- ◆ Designed to connect power, ground, and wells by abutment

February 8, 2006

<http://csg.csail.mit.edu/6.375/>

L01-25

# Standard Cell Design Examples



Channel routing for 1.0mm 2-metal stdcells

Over cell routing for 0.18mm 6-metal stdcells

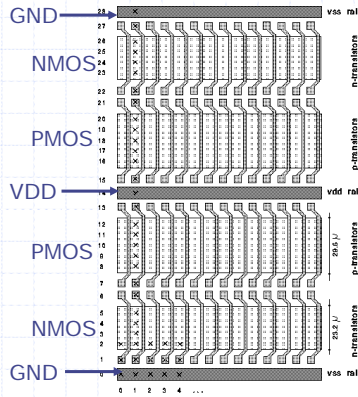
February

[csg.csail.mit.edu/6.375/](http://csg.csail.mit.edu/6.375/)

L01-26

# Gate Arrays

- ◆ Can cut mask costs by prefabricating arrays of fixed size transistors on wafers
- ◆ Only customize metal layer for each design



Two kinds:

- Channeled Gate Arrays
  - Leave space between rows of transistors for routing
- Sea-of-Gates
  - Route over the top of unused transistors

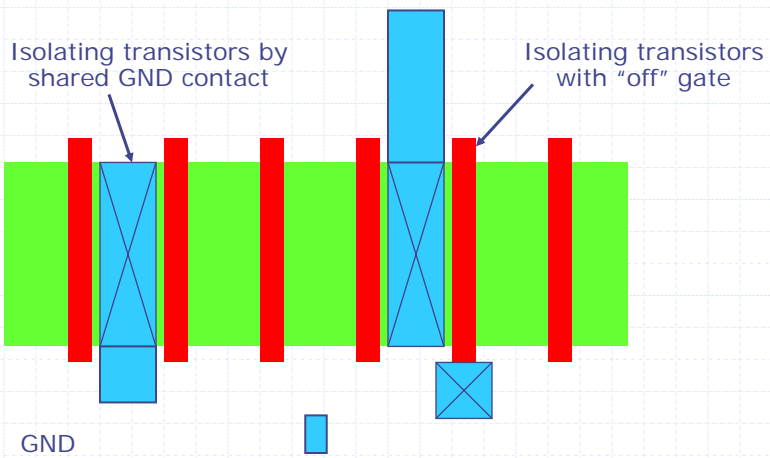
[ OCEAN Sea-of-Gates Base Pattern ]

February 8, 2006

<http://csg.csail.mit.edu/6.375/>

L01-27

# Gate Array Personalization



February 8, 2006

<http://csg.csail.mit.edu/6.375/>

L01-28

## Gate Array Pros and Cons

- ◆ Cheaper and quicker since less masks to make
  - Can stockpile wafers with diffusion and poly finished
- ◆ Memory inefficient when made from gate array
  - Embedded gate arrays add multiple fixed memory blocks to improve density (=>Structured ASICs)
  - Cell-based array designed to provide efficient memory cell (6 transistors in basic cell)
- ◆ Logic slow and big due to fixed transistors and wiring overhead
  - Advanced cell-based arrays hardwire logic functions (NANDs/NORs/LUTs) which are personalized with metal

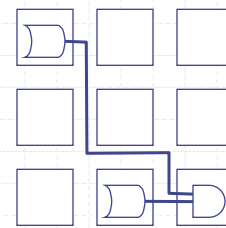
February 8, 2006

<http://csg.csail.mit.edu/6.375/>

L01-29

## Field-Programmable Gate Arrays

- ◆ Each cell in array contains a programmable logic function
- ◆ Array has programmable interconnect between logic functions
- ◆ Arrays mass-produced and programmed by customer after fabrication
  - Can be programmed by blowing fuses, loading SRAM bits, or loading FLASH memory
- ◆ Overhead of programmability makes arrays expensive and slow but startup costs are low, so much cheaper than ASIC for small volumes

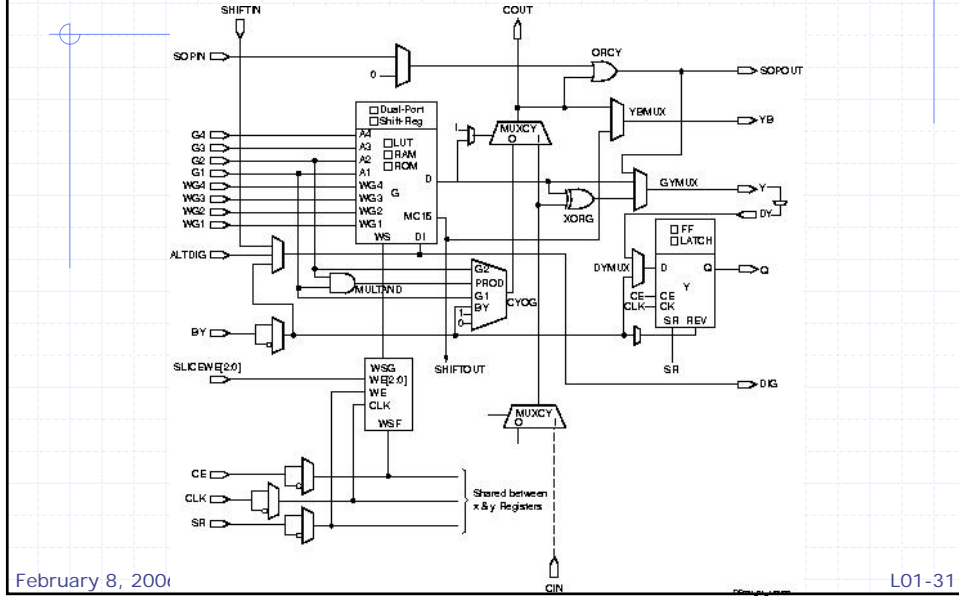


February 8, 2006

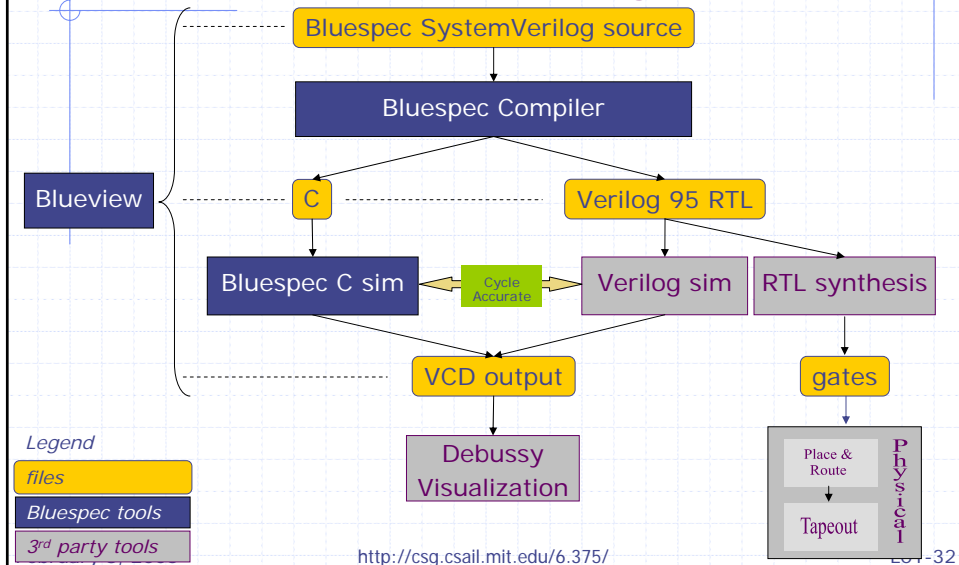
<http://csg.csail.mit.edu/6.375/>

L01-30

# Xilinx Configurable Logic Block



## 6.375 ASIC/FPGA Design Flow





## 6.375 Course Philosophy

*Design* is central focus

- Architectural design has biggest impact on development cost and final quality
- Good tools support design space exploration
  - ◆ e.g., Bluespec
- Good design discipline avoids bad design points
  - ◆ Unit-Transaction Level design to decompose upper levels of design hierarchy
  - ◆ “Best-Practice” microarchitectural techniques within units

February 8, 2006

<http://csg.csail.mit.edu/6.375/>

L01-33

## 6.375 Objectives

*By end of term, you should be able to:*

- ◆ Select appropriate implementation technology and tool flow:
  - custom, cell or structured ASIC, ASSP, or FPGA
- ◆ Decompose system requirements into a hierarchy of sub-units that are easy to specify, implement, and verify
- ◆ Develop efficient verification and test plans
- ◆ Select appropriate microarchitectures for a unit and perform microarchitectural exploration to meet price, performance, and power goals
- ◆ Use industry-standard tool flows
- ◆ Complete a working million gate chip design!
- ◆ *plan making millions at a new chip startup*

(Don't forget your alma mater!)

February 8, 2006

<http://csg.csail.mit.edu/6.375/>

L01-34

## 6.375 Prerequisites

- ◆ You must be familiar with undergraduate (6.004) logic design :
  - Combinational and sequential logic design
  - Dynamic Discipline (clocking, setup and hold)
  - Finite State Machine design
  - Binary arithmetic and other encodings
  - Simple pipelining
  - ROMs/RAMs/register files
- ◆ Additional circuit knowledge (6.002, 6.374) useful but not vital
- ◆ Architecture knowledge (6.823) helpful for projects

February 8, 2006

<http://csg.csail.mit.edu/6.375/>

L01-35

## 6.375 Structure

- ◆ First half of term (before Spring Break)
  - Lecture or tutorial MWF, 2:30pm to 4:00pm in 32-124
  - Four labs (on Athena, lab machines in 38-301)
  - Form project teams (2-3 students); prepare project proposal
  - Closed-book 90 minute quiz (Friday before Spring Break)
- ◆ Second half of term (after Spring Break)
  - Weekly project milestones, with 1-2 page report
  - Weekly project meeting with the instructor and TAs
  - Final project presentations in last week of classes
  - Final project report (~15-20 pages) due May 17 (no extensions)
- ◆ Afterwards (summer+fall commitment):
  - Possibility of fabricating best projects in 180nm technology
  - Possibility of implementing designs in FPGAs

February 8, 2006

<http://csg.csail.mit.edu/6.375/>

L01-36

## 6.375 Project

(see course web page)

- ◆ Two standard projects with fixed interfaces and testbenches:
  - MIPS microprocessor, team selects a design point:
    - ◆ High performance (e.g., speculative out-of-order superscalar)
    - ◆ Low power (e.g., aggressive clock gating, power-efficient L0 caches)
    - ◆ Minimal area (e.g., heavily multiplexed byte-wide datapath, compressed instruction stream)
  - Memory system, team selects a design point
    - ◆ Cache-coherent multiprocessor
    - ◆ Power-optimized memory system
    - ◆ Streaming non-blocking cache memory system
- ◆ Custom or non standard project:
  - Group submits two-page proposal by March 17
  - C/C++/... reference implementation running by March 22
  - Examples: MP3 player, H.264 encoder, Graphics pipeline, Network processor
  - Must work in teams of 2 or 3 students

February 8, 2006

<http://csg.csail.mit.edu/6.375/>

L01-37

## 6.375 Grade Breakdown

◆ Four Labs	30%
◆ Quiz	20%
◆ Five Project milestones	25%
◆ Final project report	25%

February 8, 2006

<http://csg.csail.mit.edu/6.375/>

L01-38

## 6.375 Collaboration Policy

- ◆ We strongly encourage students to collaborate on understanding the course material, BUT:
  - Each student must turn in individual solutions to labs
  - Students must not discuss quiz contents with students who have not yet taken the quiz
  - If you're inadvertently exposed to quiz contents before the exam, by whatever means, you must immediately inform the instructors or TA