

6.375 Complex Digital System Spring 2007



Lecturers: Arvind & Krste Asanović
TAs: Myron King & Ajay Joshi
Assistant: Sally Lee

February 7, 2007

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

L01-1

Do we need more chips (ASICs)?

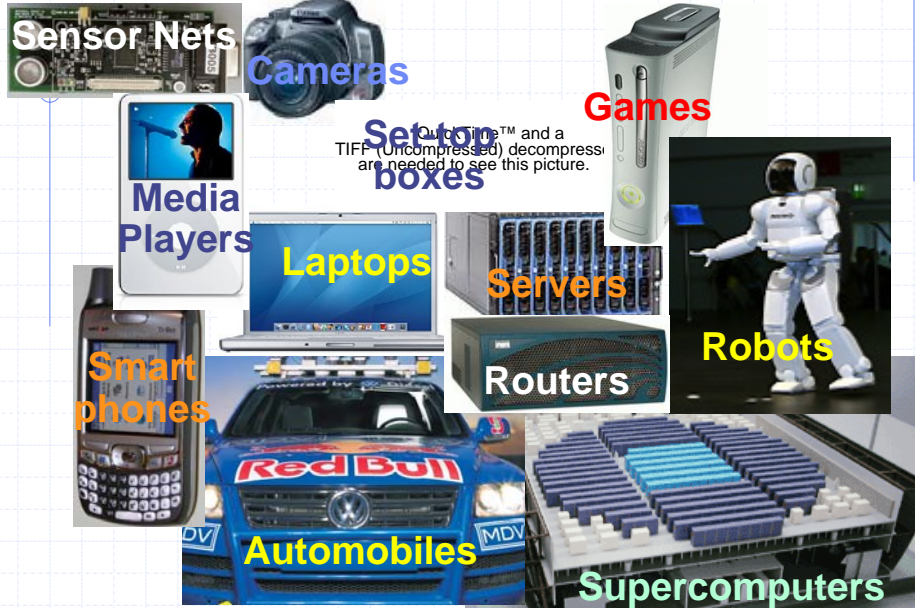
ASIC=Application-Specific Integrated Circuit

February 7, 2007

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

L01-2

Wide Variety of Products Rely on ASICs



QuickTime™ and a TIFF (Uncompressed) decompressor are needed to see this picture.

February 7, 2007

<http://csg.csail>

3

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 7, 2007

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

L01-4

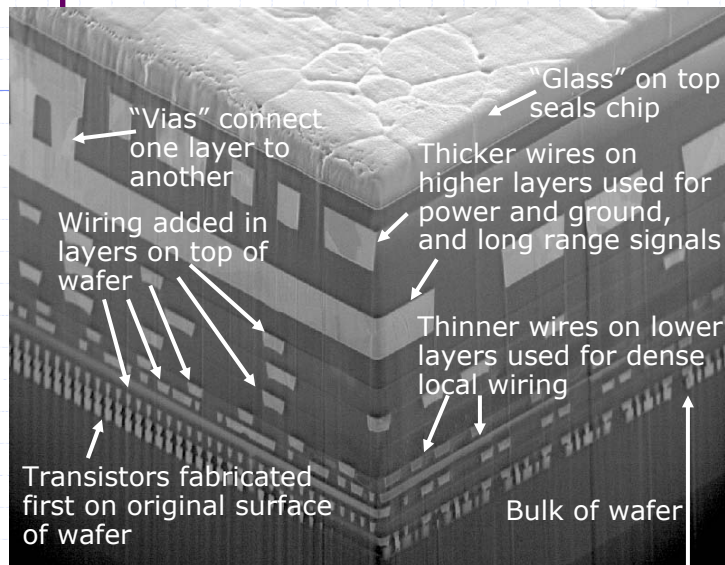
Let's take a look at current CMOS technology...

February 7, 2007

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

L01-5

Chip = Transistors + Wires



Cross-section through IBM 90nm process, 10 metal layers

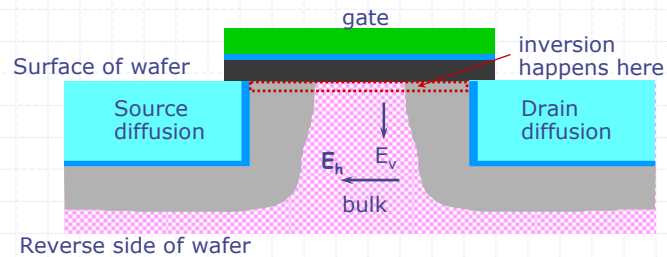
February 7, 2007

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

[ISSCC 2004] L01-6

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 7, 2007

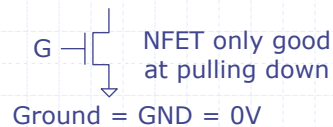
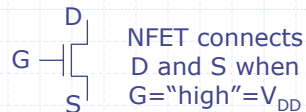
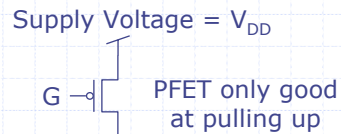
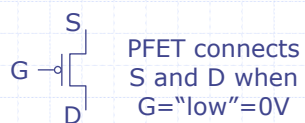
<http://csg.csail.mit.edu/6.375/>

L01-7

Simplified FET Model

Binary logic values represented by voltages:

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

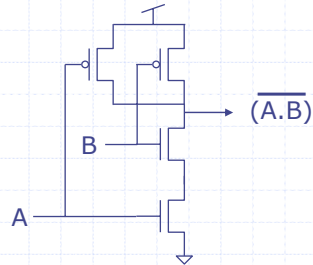
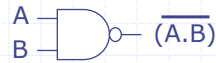


February 7, 2007

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

L01-8

NAND Gate



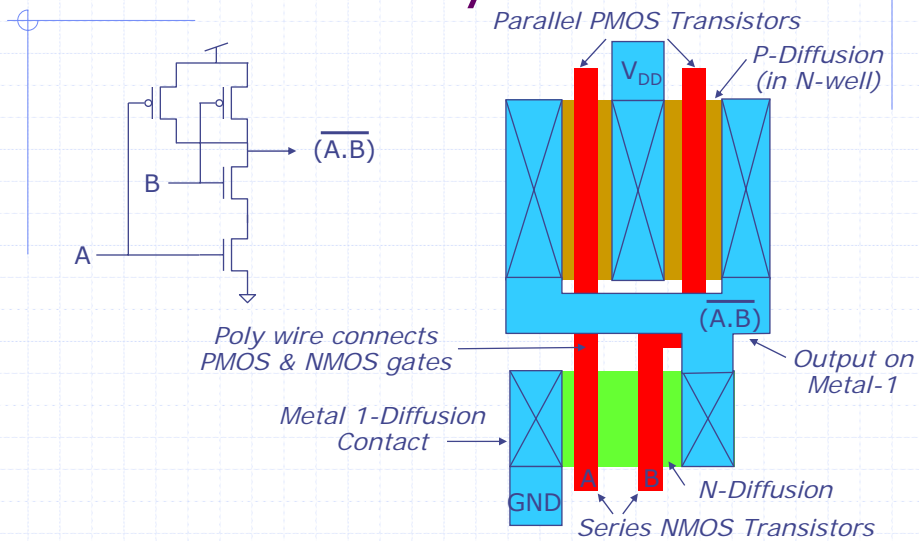
- When both A and B are high, output is low
- When either A or B is low, output is high

February 7, 2007

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

L01-9

NAND Gate Layout

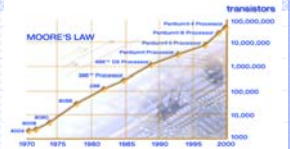


February 7, 2007

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

L01-10

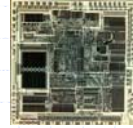
Exponential growth: Moore's Law



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



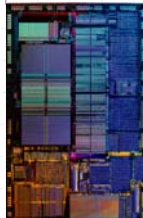
Intel 8086, 1978, 33mm²
10Mhz, 29K transistors, 3u



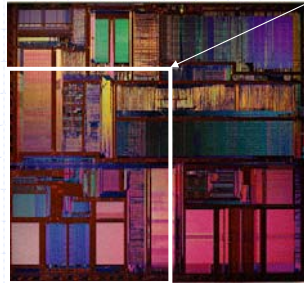
Intel 80286, 1982, 47mm²
12.5Mhz, 134K transistors, 1.5u



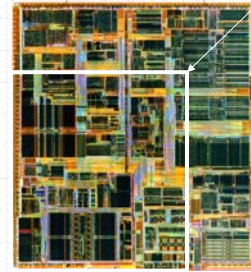
Intel 386DX, 1985, 43mm²
33Mhz, 275K transistors, 1u



Intel 486, 1989, 81mm²
50Mhz, 1.2M transistors, .8u



Intel Pentium, 1993/1994/1996, 295/147/90mm²
66Mhz, 3.1M transistors, .8u/.6u/.35u



Intel Pentium II, 1997, 203mm²/104mm²
300/333Mhz, 7.5M transistors, .35u/.25u

Shown with approximate relative sizes

http://www.intel.com/intel/museum/exhibit/hist_micro/hof/hof_main.htm

February 7, 2007

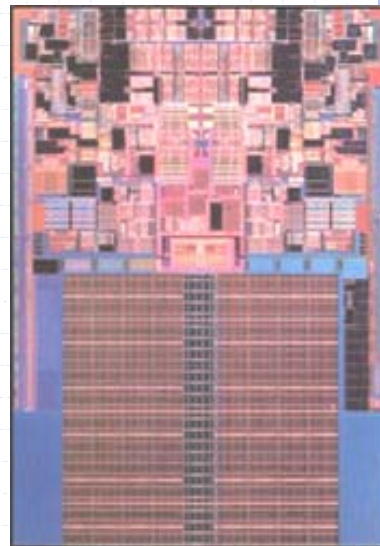
<http://csg.csail.mit.edu/6.375/>

L01-11

Intel Penryn (2007)

- ◆ Dual core
- ◆ Quad-issue out-of-order superscalar processors
- ◆ 6MB shared L2 cache
- ◆ 45nm technology
 - Metal gate transistors
 - High-K gate dielectric
- ◆ 410 Million transistors
- ◆ 3+? GHz clock frequency

*Could fit over 500 486 processors
on same size die.*

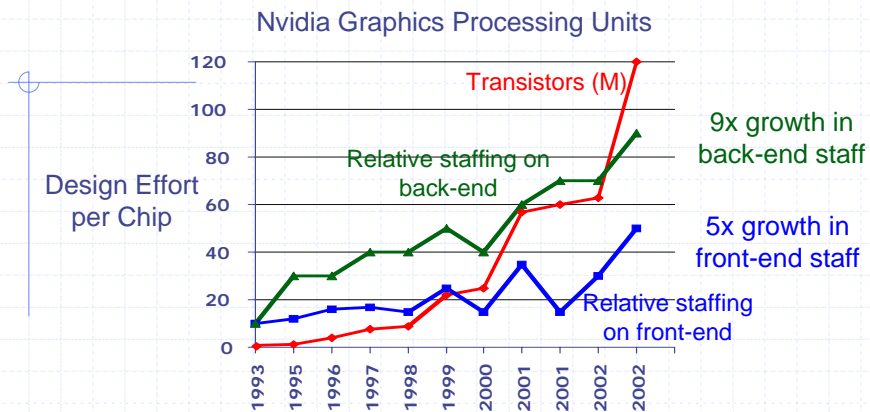


February 7, 2007

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

L01-12

.. But Design Effort Growing



- ◆ Front-end is designing the logic (RTL)
- ◆ Back-end is fitting all the gates and wires on the chip; meeting timing specifications; wiring up power, ground, and clock

February 7, 2007

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

L01-13

Design Cost Impacts Chip Cost

- ◆ 90nm ASIC cost breakdown, \$30M total (Altera study):
 - 59% chip design (architecture, logic & I/O design, product & test engineering)
 - 30% software and applications development
 - 11% prototyping (masks, wafers, boards)
- ◆ If we sell 100,000 units, Non-Recurring Engineering (NRE) costs add $\$30\text{M}/100\text{K} = \300 per chip!
- ◆ Example above is for design using automated tools
 - Similar to what we'll be using in 6.375
- ◆ Hand-crafted IBM-Sony-Toshiba Cell microprocessor achieves 4GHz in 90nm, but development cost was $>\$400\text{M}$

February 7, 2007

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

L01-14

Topics to address in 6.375

- ◆ How can we design complex billion transistor ASICs with reasonable effort?
- ◆ How good are our designs?
 - Performance, area, power

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

DM Pipeline Example	Area (pts)	Speed (ns)	Memory Util (%)
Static	8.898	3.60	63.5
Linear	4.910	4.90	99.9
Circular	4.910	3.67	99.9
Static (2)	8.898	3.60	63.5

[ICCAD'04]

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

Sub-optimal implementations!

Alternatives?

6.375 Course Philosophy

- ◆ Effective abstractions to reduce design effort
 - High-level design language rather than logic gates
 - Control specified with Guarded Atomic Actions rather than with finite state machines
 - Guarded module interfaces automatically ensure correctness of composition of existing modules
- ◆ Design discipline to avoid bad design points
 - Decoupled units rather than tightly coupled state machines
- ◆ Design space exploration to find good designs
 - Architecture choice has largest impact on solution quality

A unified view of languages, disciplines and tools that supports rapid design space exploration to find best area, power, and performance point with reduced design effort

February 7, 2007

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

L01-17

6.375 Objectives

By end of term, you should be able to:

- ◆ Decompose system requirements into a hierarchy of sub-units that are easy to specify, implement, and verify, and which can be *reused*
- ◆ 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!
- ◆ *Make millions \$\$\$ at a new chip startup*
(Don't forget your alma mater!)

February 7, 2007

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

L01-18

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 7, 2007

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

L01-19

6.375 Structure

- ◆ First half of term (before Spring Break)
 - Lecture or tutorial MWF, 2:30pm to 4:00pm in 32-124
 - Three labs (on Athena, lab machines in 38-301)
 - Form project teams (2-3 students); prepare project proposal (watch website for project ideas)
 - 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 instructors and TAs
 - Final project presentations in last week of classes
 - Final project report (~15-20 pages) due Thursday May 17 (no extensions)

February 7, 2007

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

L01-20

6.375 Grade Breakdown

◆ Three labs	30%
◆ Quiz	20%
◆ Five project milestones	25%
◆ Final project report (including presentation)	25%

February 7, 2007

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

L01-21

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

February 7, 2007

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

L01-22

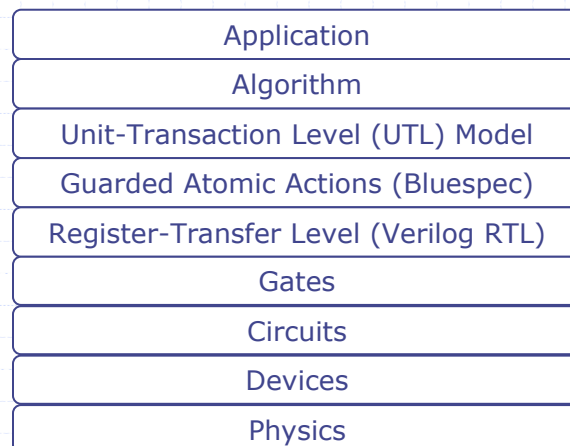
ASIC Design Styles

February 7, 2007

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

L01-23

Hardware Design Abstraction Levels



February 7, 2007

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

L01-24

ASIC Design Styles

- ◆ Full-Custom (every transistor hand-drawn)
 - Best possible performance: as used by Intel μ Ps
- ◆ Semi-Custom (Some custom + some cell-based design)
 - Reduced design effort: AMD μ Ps plus recent Intel μ Ps
- ◆ Cell-Based ASICs (Only use cells in standard library)
 - High-volume, moderate performance: Graphics chips, network chips, cellphone chips
 - This is what we'll use in 6.375
- ◆ Mask-Programmed Gate Arrays/Structured ASICs
 - Medium-volume, moderate performance applications
- ◆ Field-Programmable Gate Arrays
 - Low-volume, low-moderate performance applications, and prototyping
- ◆ Comparing styles:
 - how many design-specific mask layers per ASIC?
 - how much freedom to develop own circuits?
 - what design methods and tools are needed?

February 7, 2007

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

L01-25

Custom and Semi-Custom

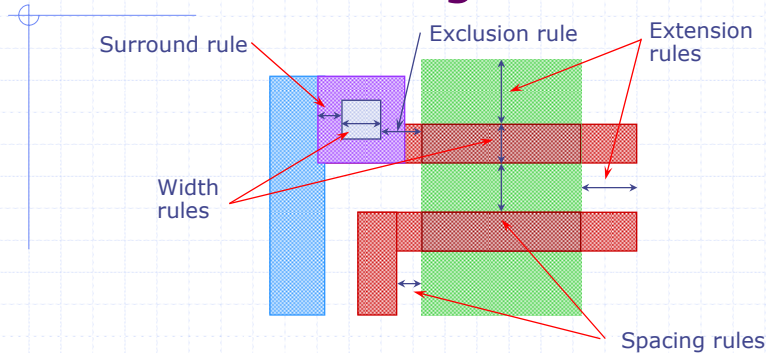
- ◆ Usually, in-house design team develops own libraries of cells for commonly used components:
 - memories
 - register files
 - datapath cells
 - random logic cells
 - repeaters
 - clock buffers
 - I/O pads
- ◆ In extreme cases, every transistor instance can be individually sized (\$\$\$\$)
 - approach 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 7, 2007

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

L01-26

Custom Designer works with Low-Level 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 7, 2007

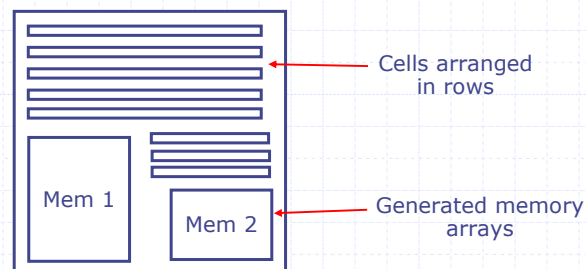
<http://csg.csail.mit.edu/6.375/>

L01-27

Standard Cell ASICs

aka Cell-Based ICs (CBICs)

- ◆ Fixed library of cells + memory generators, often provided by fabrication foundry or third-party library providers
- ◆ 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)

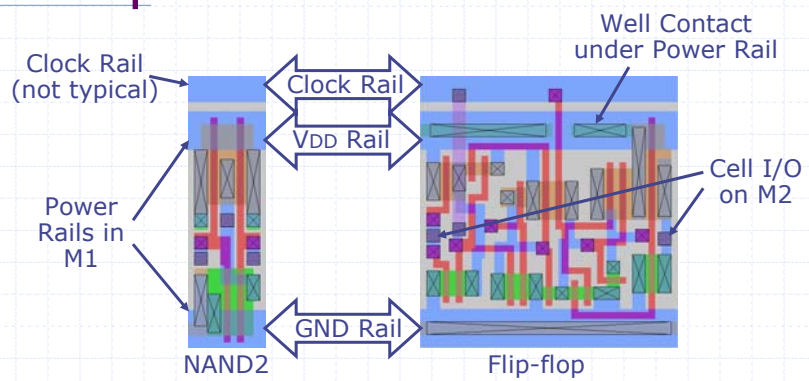


February 7, 2007

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

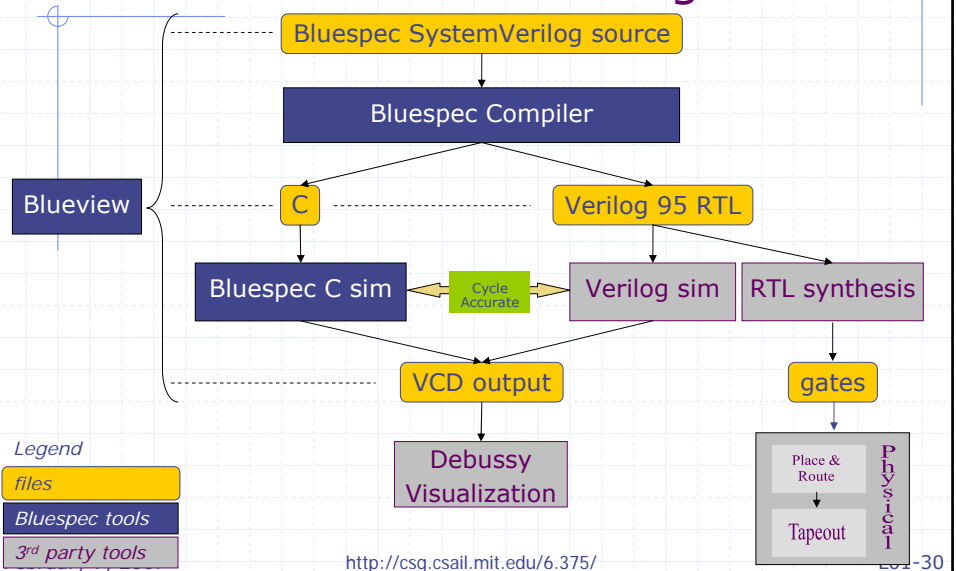
L01-28

Standard Cell Library Components



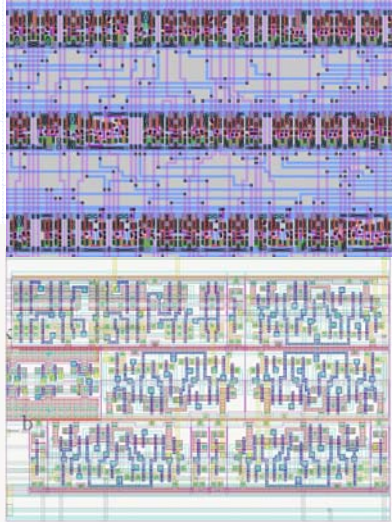
- ◆ Cells have standard height but vary in width
- ◆ Designed to connect power, ground, and wells by abutment

6.375 Standard Cell Design Flow



- Legend
- files
 - Bluespec tools
 - 3rd party tools

Standard Cell Design Examples



Channel routing for 1.0mm 2-metal stdcells

Over cell routing for 0.18mm 6-metal stdcells

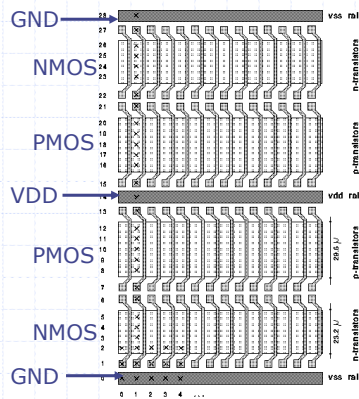
February

mit.edu/6.375/

L01-31

Mask-Programmed 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

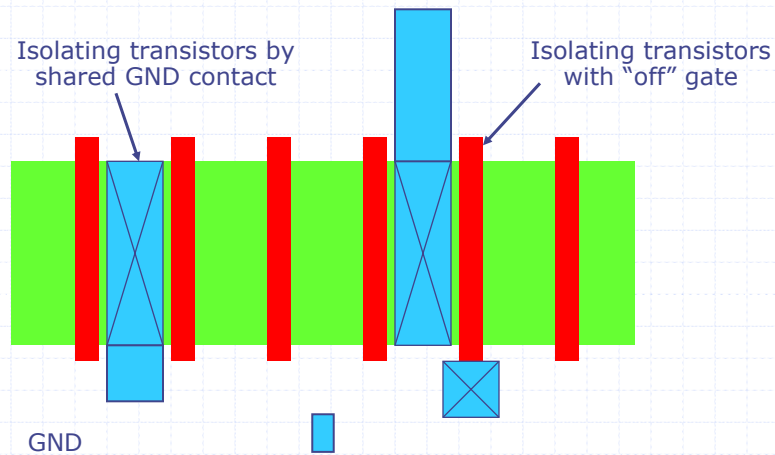
[OCEAN Sea-of-Gates Base Pattern]

February 7, 2007

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

L01-32

Gate Array Personalization



February 7, 2007

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

L01-33

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

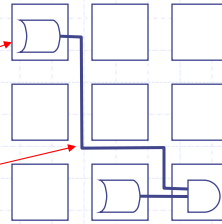
February 7, 2007

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

L01-34

Field-Programmable Gate Arrays (FPGAs)

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

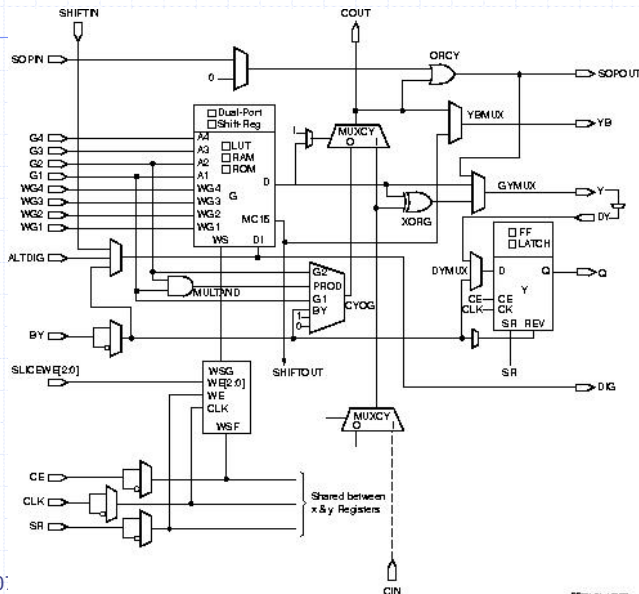


February 7, 2007

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

L01-35

Xilinx Configurable Logic Block



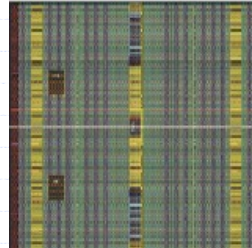
February 7, 2007

L01-36

FPGA Pros and Cons

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 $\sim 12X$ worse
- Performance up 3-4X worse
- Area 20-40X greater

Still requires tremendous design effort at RTL level