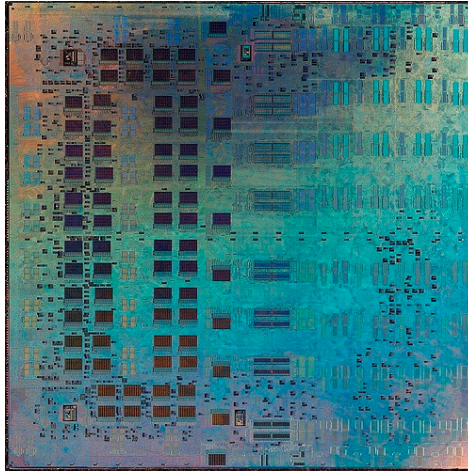

Computer Hardware Design

EECS 4340



Prof. Simha Sethumadhavan

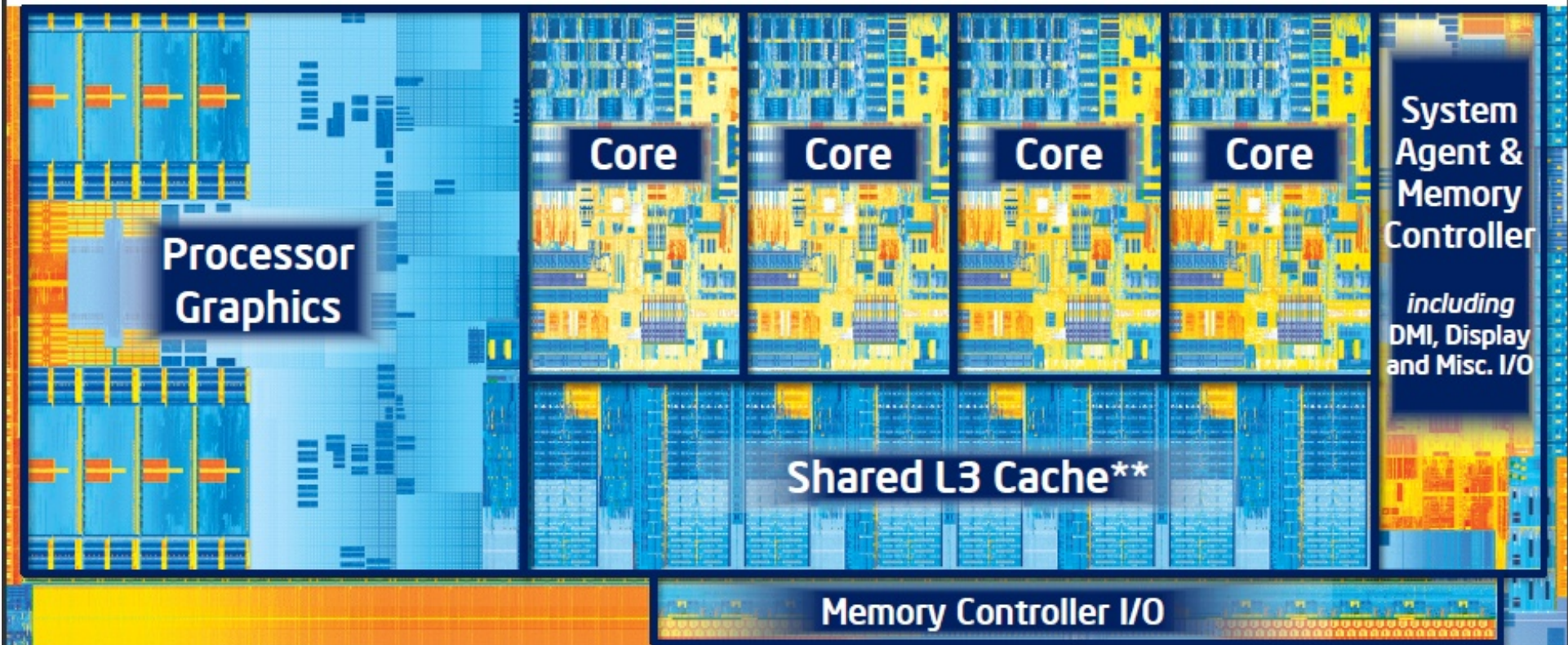
Course Description

- **Practicum on hardware design**
 - *“A college course, often in a specialized field of study, that is designed to give students supervised practical application of a previously or concurrently studied theory.”*
- **Theory: Understand hardware design flow**
 - From initial planning through all engineering steps to tapeout
 - Use what you learned in prior hardware & programming classes
- **Practice: Build hardware**
 - You will use state-of-the-art commercial tools
 - Lectures will cover technology behind some of these tools
- **Supervision: I will emphasize on**
 - Being professional & thorough (See projects from last time)
 - Rigorous, industrial-strength, random validation
 - **Philosophy: Your design is wrong. And, less wrong after validation.**
- **Class time: 25% theory, 25% tools, and 50% project!**

Hardware Design Experience is Valuable

- Everyone needs hardware
 - Hardware is the foundation for all modern IT
 - Hardware design engineers are employed (at:) AMD, ARM, Apple, Boeing, Broadcom, Cavium, Cray, Cisco, Dell, D.E. Shaw, Fujitsu, Freescale, Hewlett-Packard, Hitachi, Lockheed-Martin, Intel, IBM, Motorola, Nvidia, Northrop Grumman, Oracle, Phillips, Raytheon, Qualcomm, Samsung, Synopsys, Texas Instruments, Toshiba etc., and many startups.
- Learn principles for engineering billion component systems
 - Engineering discipline like no other! Understand how to manage complexity.
- Helps you design better software
 - Understand hardware trends
 - Future software must match the abilities of future hardware
 - Personal observation: My software engineering skills improved as a result of my hardware engineering experience
 - Consider this: Mozilla code base is roughly the same size as openSPARC T1 but compare number of bugs!

3rd Generation Intel® Core™ Processor: 22nm Process

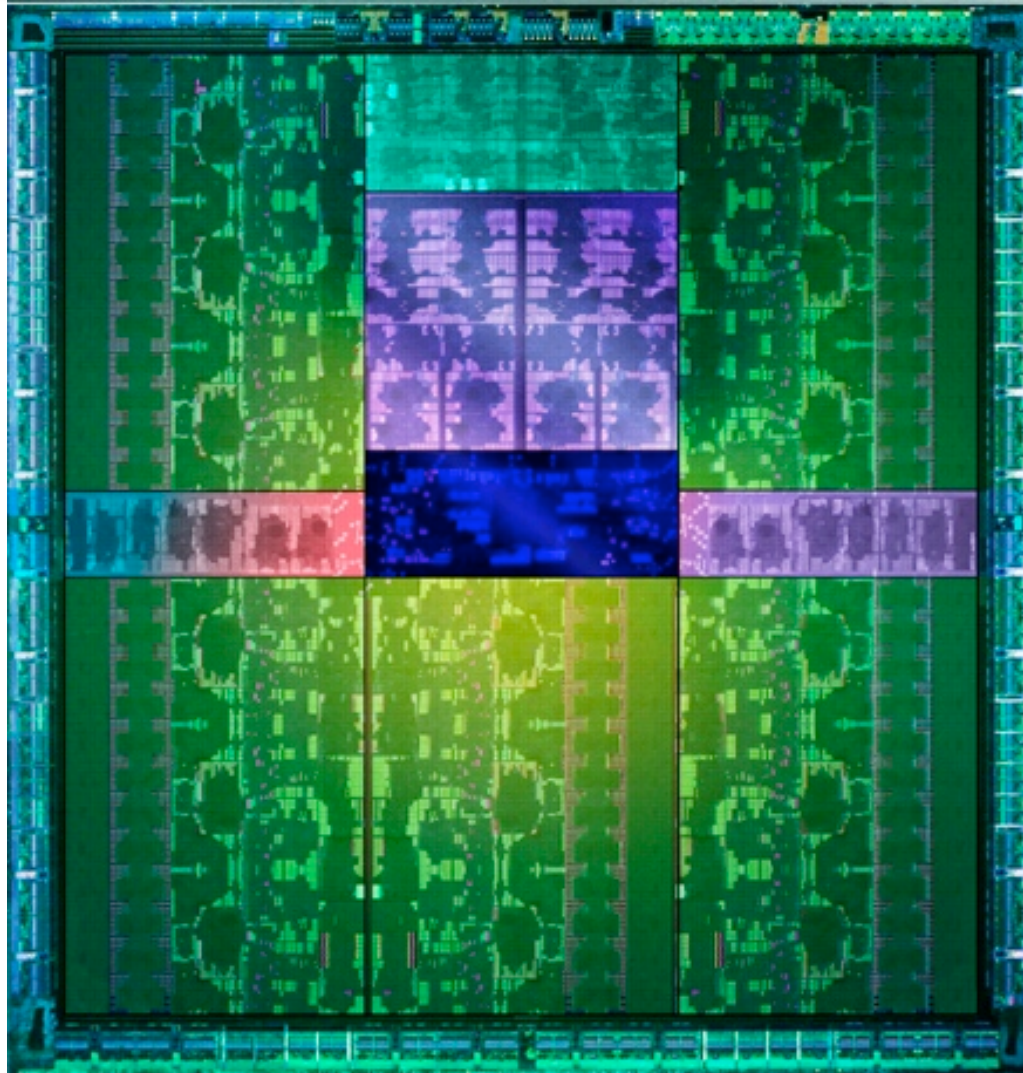


New architecture with shared cache delivering more performance and energy efficiency

Quad Core die with Intel® HD Graphics 4000 shown above
Transistor count: 1.4Billion Die size: 160mm²

** Cache is shared across all 4 cores and processor graphics

NViDIA (7.1 Billion Transistors)



Course Logistics

- **Mailing list:**
 - 4340_fa12@lists.cs.columbia.edu
 - Email: simha@cs.columbia.edu
- **Lectures on Tuesdays and Thursdays**
 - Second half of the class (starting Nov 2) dedicated to the project
- **Office hours: Mon 5:00 – 6:00; Wed 5:00 – 6:00**
- **Workload**
 - Two labs (10 and 10%), roughly 14 days per lab, 3 person teams
 - Midterm on 10/25 (in class) - 20%
 - Final project on 12/20 – 60%
 - Two standard projects, work in groups of 4, open to student projects
 - Stay on schedule! Graded on intermediate milestones
 - Possibility of fabrication – 8/12 month commitment afterwards
- **Labs**
 - We will use the CS CLIC lab. TA will be available for help.
 - If you are not in CS you should apply for an account - \$50

A Hardware Design Engineer Must...

- understand what drives the field...
 - Moore's law
- ... convert copious raw transistors into products ...
 - design principles
- ... that function correctly ...
 - validation, testing techniques
- ... and maximize profit.
 - Understand time-to-market and choose best perf/time
- THIS LECTURE: Overview of all these aspects.

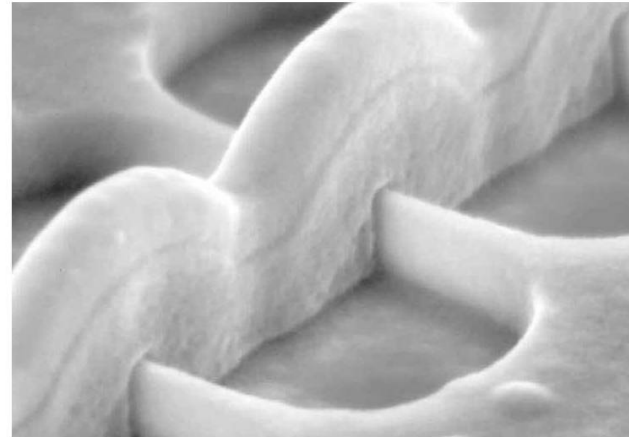
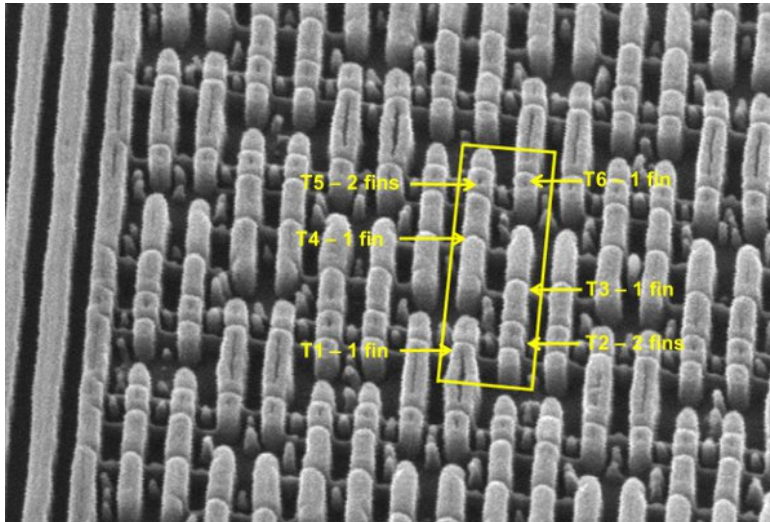
Moore's law

- 1965: “The complexity of Integrated Circuits for minimum component costs has increased at a rate of roughly a factor of two per year” [c.f. Cramming More Components onto Integrated Circuits] “*per year doubling*”
 - Complexity is defined as the number of components per chip.
- 1975: Retrospective from Moore [c.f. Progress in Digital Integrated Electronics] “*doubling every two years*”
 - ↑Chip Area, ↓ Feature Size, ↑Device and Circuit Designs
 - Revised predictions to doubling every two years
- 1990s: ITRS became industry standard for Moore's law predictions. “*doubling every 18 months*”

Moore's law

- 1965: “The complexity of Integrated Circuits for minimum component costs has increased at a rate of roughly a factor of two per year” [c.f. Cramming More Components onto Integrated Circuits] “*per year doubling*”
 - Complexity is defined as the number of components per chip.
- 1975: Retrospective from Moore [c.f. Progress in Digital Integrated Electronics] “*doubling every two years*”
 - ↑Chip Area, ↓ Feature Size, ↑Device and Circuit Designs
 - Revised predictions to doubling every two years
- 1990s: ITRS became industry standard for Moore's law predictions. “*doubling every 18 months*”

Fabrication: 10,000 ft Overview



Behind Moore's Law: Process Scaling

- Feature size scaling: Shrinking the physical size of the transistors and the wires interconnecting them.
- Benefits:
 - Increased functionality in the same area
 - more devices on a chip => more complex functions can be implemented
 - or same functionality in a smaller area footprint
 - Smaller chip => more dies per wafer => more profit per wafer
 - Further, smaller devices are faster
 - And, consume less energy to operate!
- Process scaling allows chips that provide more performance and functionality, and therefore sell more, and to be manufactured at a lower cost.

Process Scaling Projections (ITRS 10)



Parameters:

- Feature Size: $0.7x \Rightarrow \text{Area} = 0.5x$
- Capacitance(C): $0.6x$
- Supply voltage(V_{dd}): $0.9x$
- Power (CV^2F): $\sim 0.5 \cdot Fx$
- Power density should remain constant
- $\Rightarrow$ Frequency: $1.0x$

Scaling Projections (Industry)

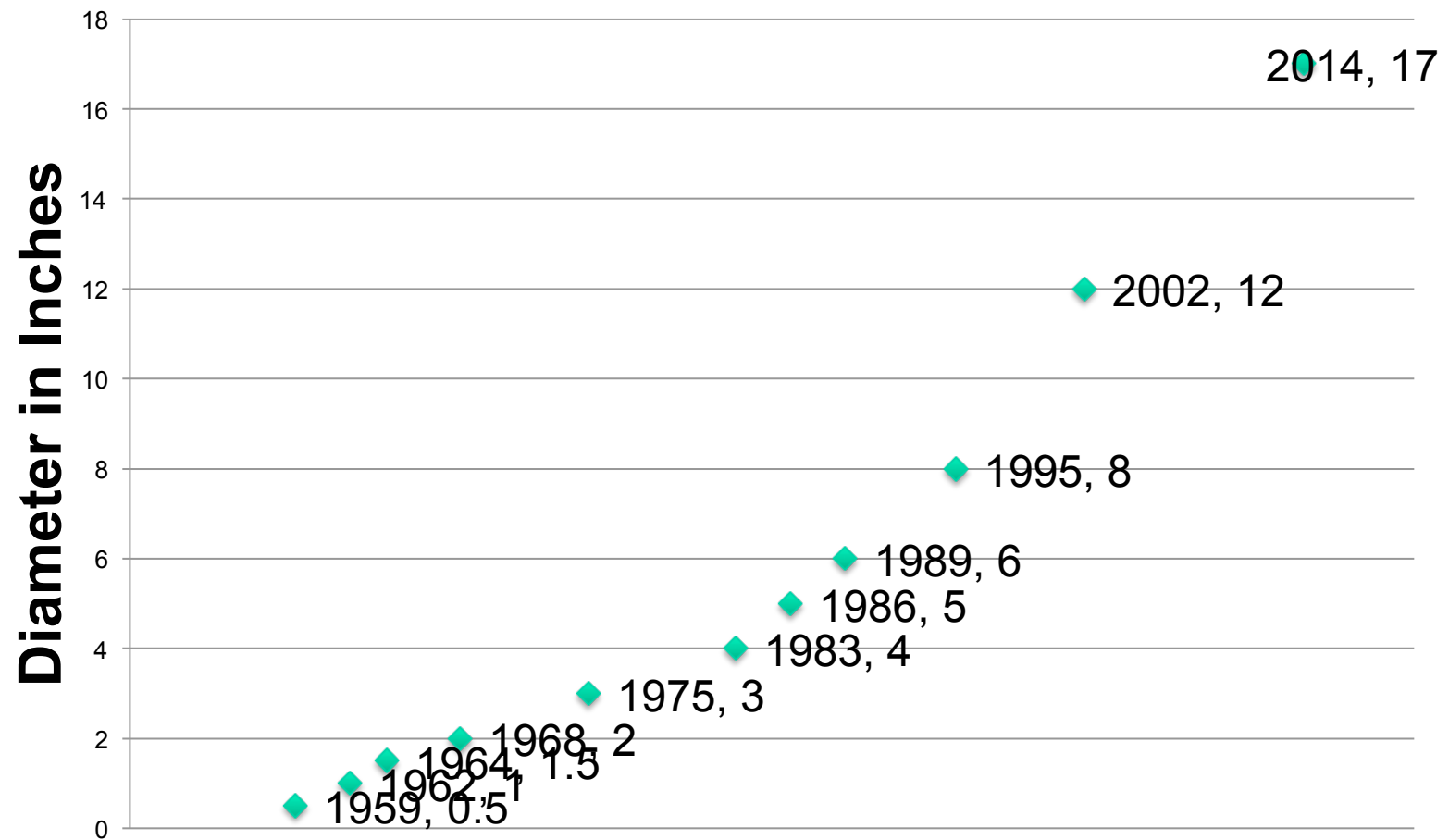
Node (nm)	Feature Size	Area	Cap.	Freq.	V _{dd}	Power	Power Density
45 to 32	0.75x	0.57x	0.66x	1.10x	0.925x	0.62x	1.09x
32 to 22	0.75x	0.57x	0.66x	1.08x	0.950x	0.64x	1.13x
22 to 14	0.75x	0.57x	0.66x	1.05x	0.975x	0.66x	1.16x
14 to 10	0.75x	0.57x	0.66x	1.04x	0.985x	0.61x	1.17x

Adapted from:

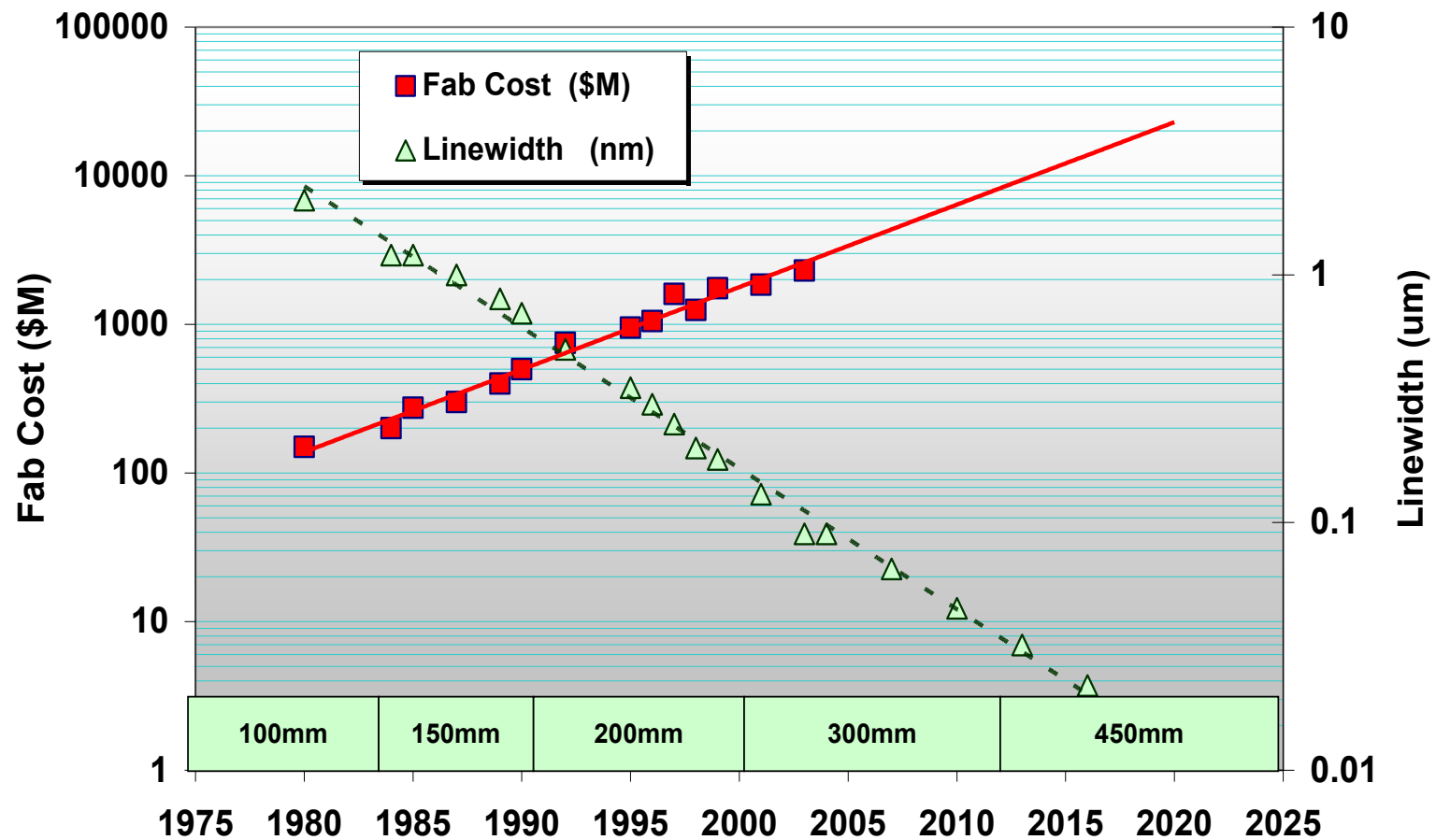
Scaling with design constraints: predicting the future of big chips (Rajamani)

The Exascale Challenge (Borkar)

Behind Moore's Law: Wafer Scaling



Moore's law Cost



SRC: ITRS Roadmap 2012

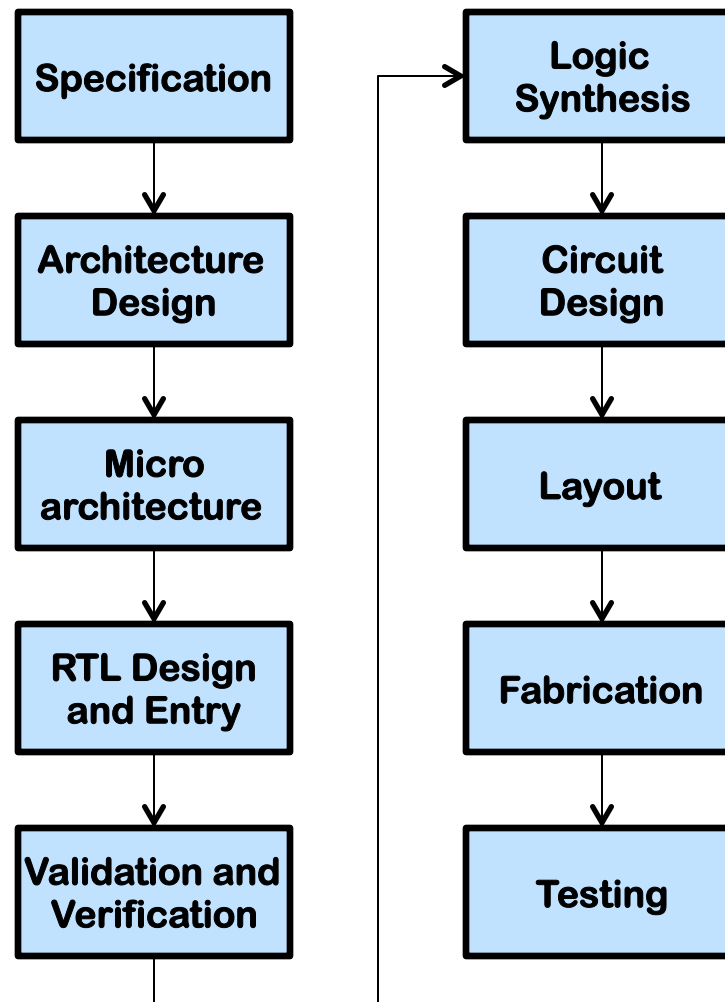
Implications

- **Moore's law will continue for at least 20+ years**
 - ~10 years of process scaling: on its last legs
 - ~10 years with one-shot improvements like 450mm wafer, 3d
 - Lithography improvements? EUV, e-beam, self-assembly etc.,
- **But have to be really smart about design**
 - Invent techniques to handle more complexity
 - Power and energy-efficiency are major concerns
 - Make this a zeroth-order design requirement
 - Wires are not scaling
 - Optimize for communication during design
 - Transistors are not free
 - They leak, wafer scaling limits transistors, they cost more to manufacture
 - Focus on design and architectural decisions
 - Small optimizations at lower levels are useful but give you small benefits

A Hardware design engineer must ...

- ✓ understand what drives the field...
 - ✓ Moore's law, technology trends
- ... convert copious raw transistors into products ...
 - design principles
- ... that function correctly ...
 - validation, testing techniques
- ... and maximize profit.
 - understand time-to-market and choose among options
- **THIS LECTURE: Overview of all these aspects.**

Digital Hardware Engineering Steps



A Simple Running Example

Design a very simple processor with four instructions.

goal: to add two four byte numbers and report overflow.

Architecture Design

- Overarching architecture question
How to open up the hardware to software?
- For our design: What instructions do you need?
- At least four:
 - Instruction to perform addition
 - Instruction to load inputs
 - Instruction to store results
- How are operands specified?
 - From memory? Through registers?

A Sample Architectural Specification

OVERVIEW

This processor uses the von Neumann, three operand load-store architecture for processing. The load instructions fetch four bytes of data from a 4-byte aligned memory address into register locations. The processor has four registers: three general-purpose registers and a status register. This simple processor has one arithmetic instruction; this instruction reads two source registers, adds each byte pairwise, and writes the output to a destination register. On an overflow the processor sets the bit corresponding to the overflow to 1. The processor has an conditional jump instruction. When the machine is powered up it starts fetching instruction from address 0. The machine can address locations each of which holds 1 byte of data.

What else can you have add?

A Sample Architectural Specification

Instructions and Instruction Formats

ADD

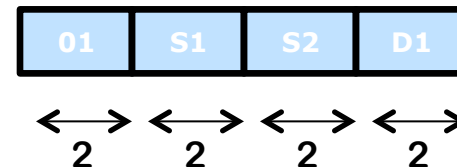
This instruction performs addition of two 32 bit values stored in register files. It takes two operands s1 and s2 and stores the result in the destination register. If any of the additions result in an overflow the value is stored in the overflow register at the byte location that caused the overflow.

The instruction opcode is 01.

Example:

[1] [1] [1] [1]
[0][0][0][0]

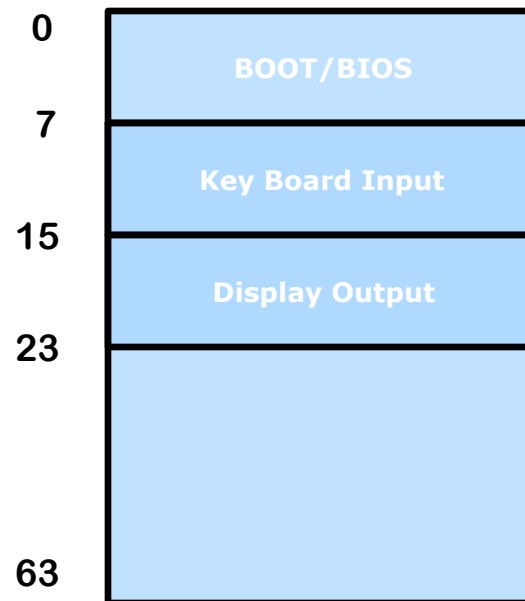
Instruction Format



A Sample Architectural Specification

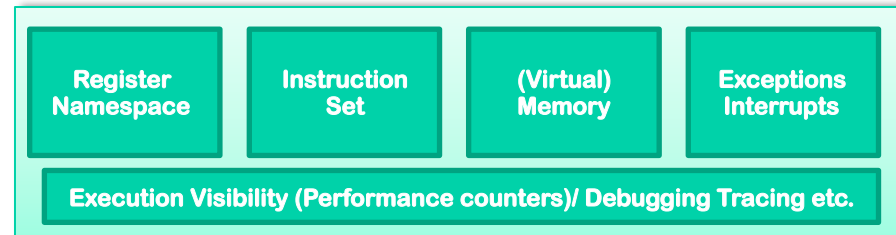
MEMORY

The processor can address 64 addressable memory locations. Each memory location holds 8 bits.



More Generally: System Architecture

- Typical features exposed through the ISA are:



- Major concern: Backward and Forward Compatibility
 - ISA extensions typical in the microprocessor world
 - New ISAs and execution models more likely in the embedded space
- Team produces a complete specification of system level architecture and defines the exact semantics of each instruction.
 - x86 manuals (<http://www.intel.com/products/processor/manuals/>)
 - See manuals passed around in class for structure

Microarchitecture

How is the ISA implemented?

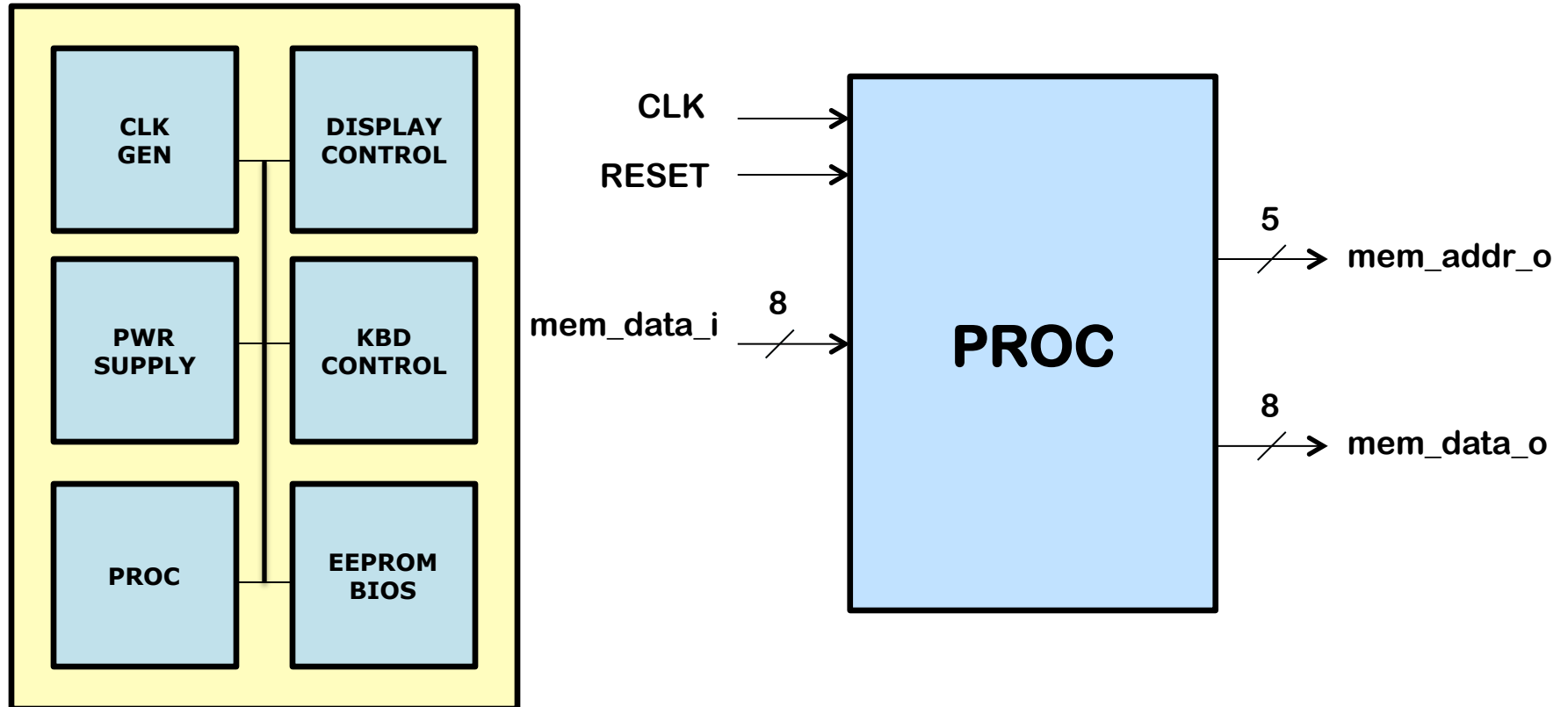
- Specify the type, granularity and organization of the units that support the ISA
- Optimize for the common case
- A microarchitectural Question for our simple processor: What are possible ways to implement the add instruction?
 - Goal: minimize area:
 - Goal: High performance

More Generally: Microarchitecture

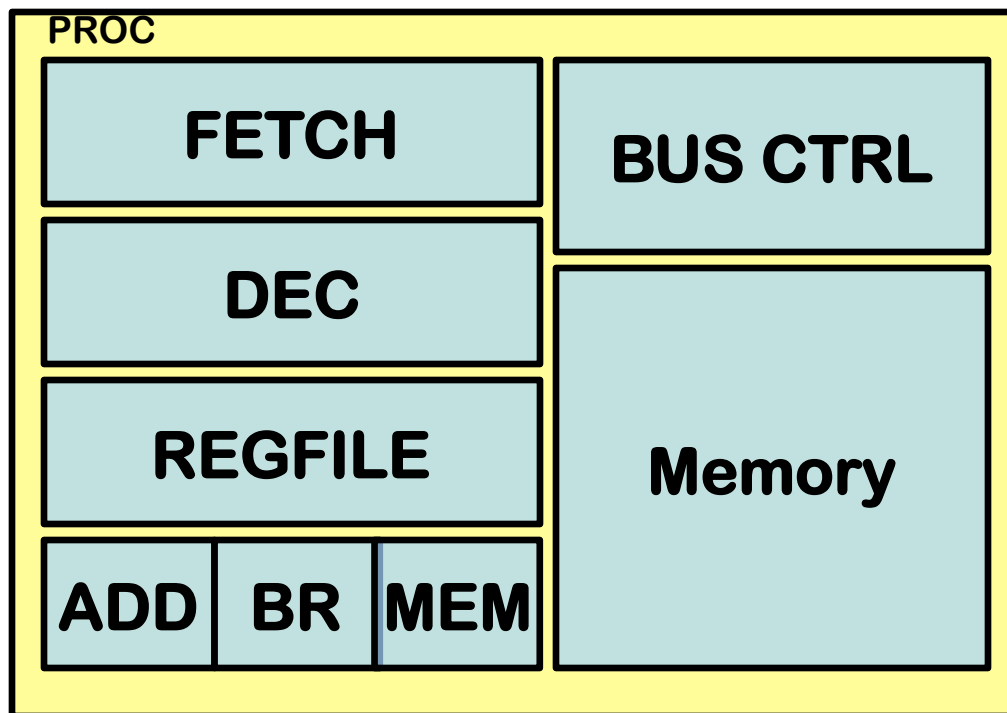
- **Microarchitectural techniques**
 - A major design differentiator
 - Allow realization of the benefits of technology improvements
 - Pipelining enabled faster clock frequencies
 - Can compensate for shortcomings of technology
 - Memory hierarchies mitigate losses due to slow, pin limited storage
- **Microarchitects use simulators to study many tradeoffs**
 - Tradeoffs: Power/Energy/Area/Performance/Temperature/Reliability
 - Many simulators today are written in software and tend to be slow
 - Can use hardware techniques to speed simulators
- **Major microarchitecture parameters are determined before design**
 - Continued minor refinement during the hardware design process

RTL Design

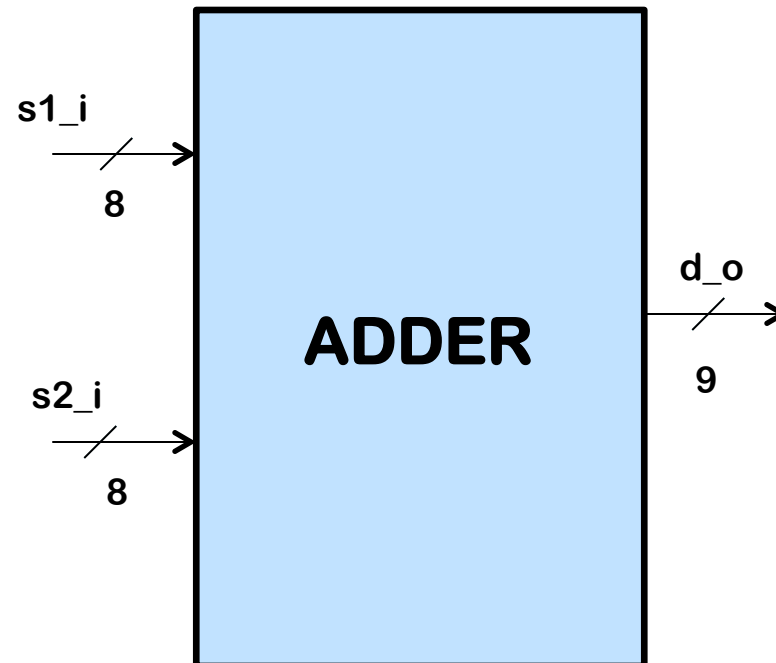
System Block Diagram



RTL Design: Unit Partitioning



Unit Specification



Signal Name	Type	Width	Description
s1_i	input	8	Input operand from register s1
s2_i	input	8	Input operand from register s2
d_o	output	9	The LSB 8 bits are the sum; the MSB is the overflow.

Register Transfer Language Entry

```
interface ifc;
    logic [7:0] s1_i;
    logic [7:0] s2_i;
    logic [8:0] d_o;
    modport add (input s1_i, s2_i, output d_o);

endinterface

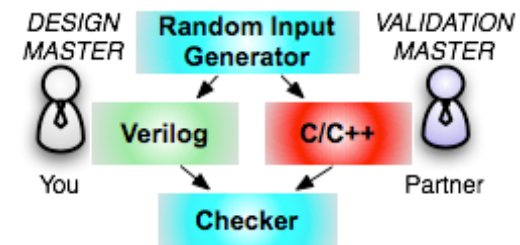
module adder(ifc.add i);
    assign i.d_o = i.s1_i + i.s2_i;
endmodule
```

More Generally, RTL Design and Entry

- Implements microarchitecture
- Steps
 - Partition the microarchitecture into a set of units
 - Fully specify the interfaces between the units (freeze)
 - Partition each unit into sub-units, and specify interfaces
 - Assign Design Master, Unit Owners, Validation Master, Unit Verifiers and Integration Master
 - Write detailed microarchitectural specifications
 - Include block diagrams for each unit/sub-unit and specify interfaces
 - Estimate timing and area (in terms of number of flip-flops/gates)
 - Specify validation methodology
 - Highlight tricky corner cases
 - Specify power/thermal management optimizations
 - Hold design review
 - Start RTL entry, start building validation infrastructure
 - Iterate until convergence, hold many more design reviews
- Check out the manual that is being passed around
- We will closely follow this for the lab and class projects

Verification & Validation

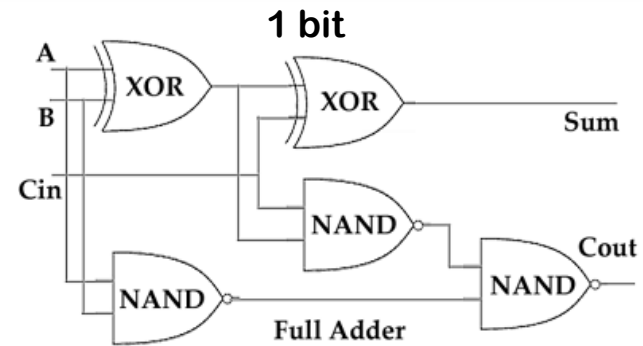
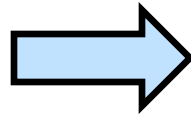
- Bugs are expensive
 - Post-tapeout bugs are catastrophic
 - In 1995 Intel recalled processors because of a bug in a floating point unit.
 - Recall cost: ½ Billion US dollars
 - Pre-tapeout bugs also hurt
 - For a microprocessor every 18 months, performance improves by ~36%
 - 2 weeks for bug fix for a new feature => 1% performance loss
 - In this class, we will
 - Understand sources of complexity
 - Learn defensive implementation techniques



- V&V is a *process* used to demonstrate that the intent of a design is preserved in its implementation.
 - Use diversity/duplication to reduce bugs
 - Use random testing

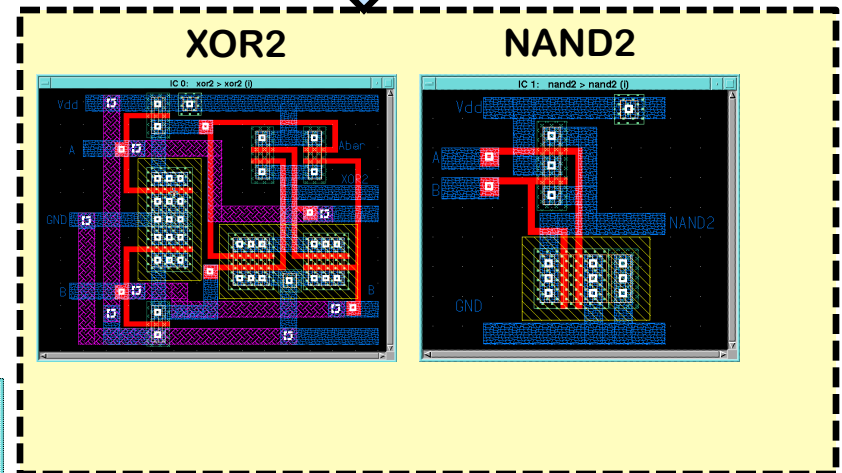
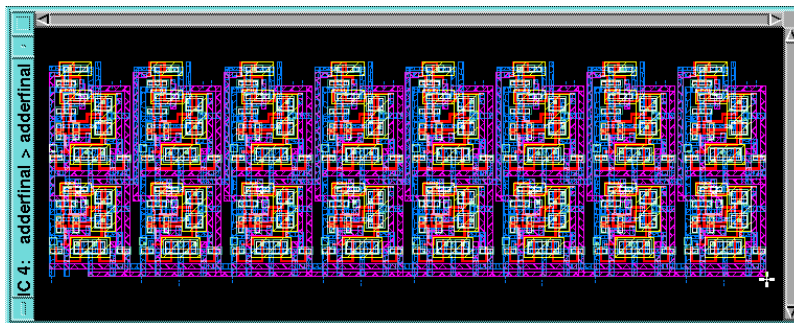
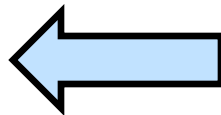
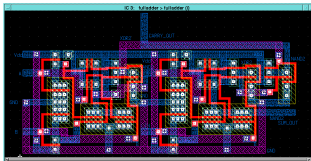
Post Validation to Circuits, Layout

```
interface ifc;  
  logic [7:0] s1_i;  
  logic [7:0] s2_i;  
  logic [8:0] d_o;  
  modport add (input s1_i, s2_i, output d_o);  
endinterface  
  
module adder(ifc.add i);  
  assign i.d_o = i.s1_i + i.s2_i;  
endmodule
```



Standard Cell Library
Or Custom cells

1 bit Adder layout



8-bit layout adder

Generally, Logic Synthesis

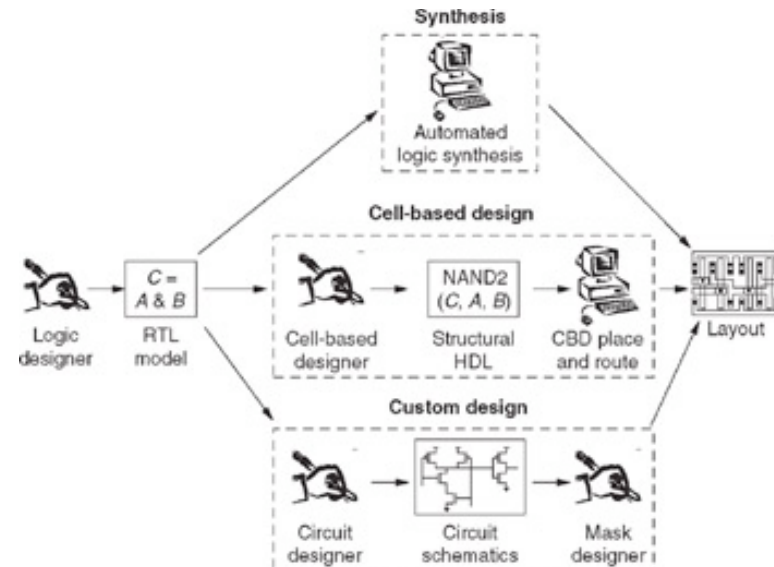
- Logic Synthesis

- Process of converting from a relatively abstract HDL model of the desired behavior to a structural model that can be realized in hardware.

- Three choices

- Automatic synthesis (this class)
- Semi-custom design
- Full-custom design

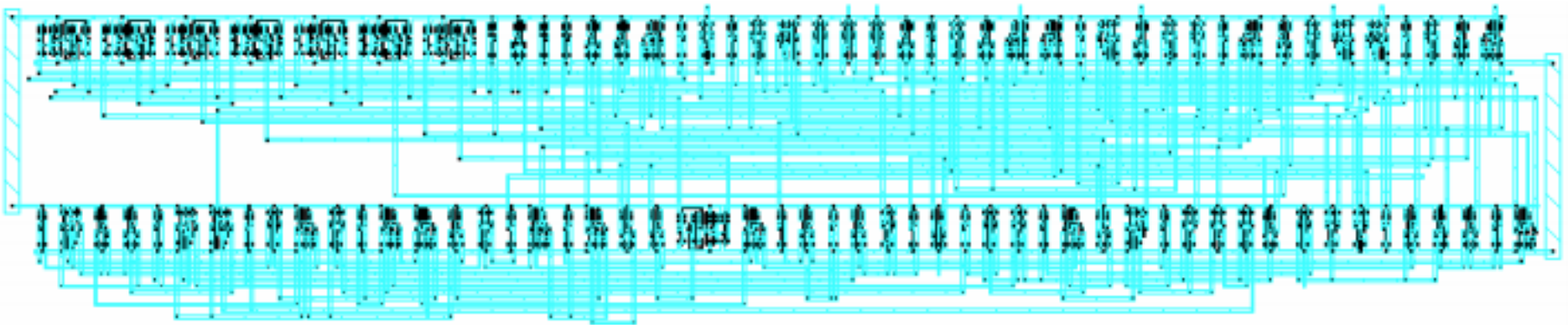
- Automation allows complexity to grow without equivalent increase in team size



Layout

- Determines the positioning of the different layers of material that make up the transistors and wires in the circuit design.
- Primary focus: “drawing” the needed circuit in the smallest area that can still be manufactured.
- Other important foci:
 - Power/CLK routing
 - Design for testability
 - Ensure that the synthesized design matches the HDL/circuit design using Layout Vs. Schematic tools (LVS)
- Significant impact on the frequency and reliability of the circuit.
- Completion of physical design is called tapeout

Place and Route using Standard Cells

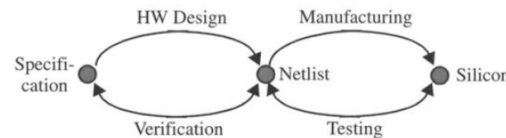


Manufacture



Silicon Debug

- **Test silicon**
 - Testing on wafers
 - Testing on dies
 - Package testing
 - Post-package testing
 - Boot a real OS!
- **Test**
 - Exercise physical locations in a chip
 - Check if they can go from 0->1 and 1->0
 - And if the change can be observed
 - We will learn tools and techniques for inserting observability



The impact of your design choices

- Many contributing factors to final cost of the product
 - Non recurring design costs
 - Non recurring fabrication costs
 - Opportunity cost due to delays – time-to-market
 - Recurring power and energy costs
- With a simple example we will explore how manufacturing cost can affect product cost
 - Wafer Cost, Wafer Yield,
 - Die Yield, Die Size
 - Packaging cost,
 - Testing cost etc.,

Simple Economics

Costs:

- **Manufacturing cost**
 - Need multiple mask layers
 - Full mask “set” costs (50/65 Masks) ~ \$5M
 - Fabs require minimum lot size ~ 6 wafers
 - Parts = 1000 – 10000 chips/min “spin”
 - Depending on size of the die
 - Raw cost/unit = mask costs/ # parts
 - For small volumes part costs could be \$5000 - \$500!
 - Lesson: Generally, more masks is better with higher volumes
- **Design cost**
 - Say, each designer on average costs \$150K/yr (loaded)
 - Design team size
 - Microprocessor ~ 400 => design cost = $400 * \$150K * \# \text{ years}$
 - Microcontrollers ~ 10 => design cost = $10 * \$150K * \# \text{ years}$
 - Time to product
 - Microprocessor ~ 4.5 years
 - Microcontroller ~ 1 year
- Design and manufacturing cost both significant contributors

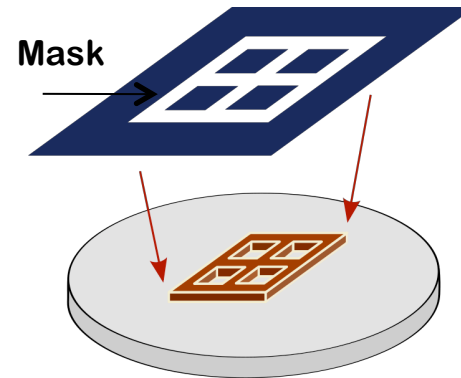
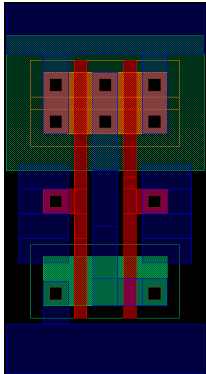


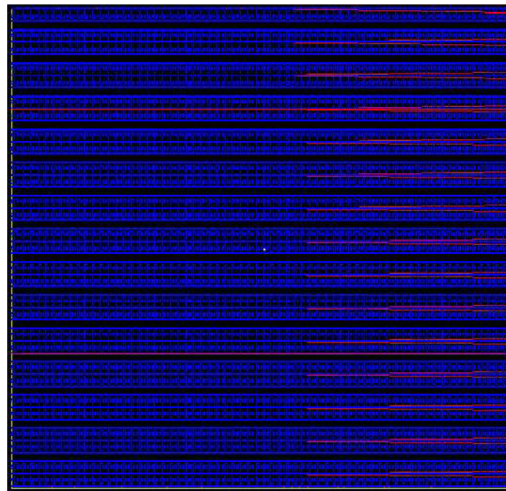
Image Src:
Wiki

Example 1: Structured ASICs

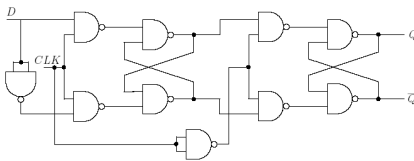
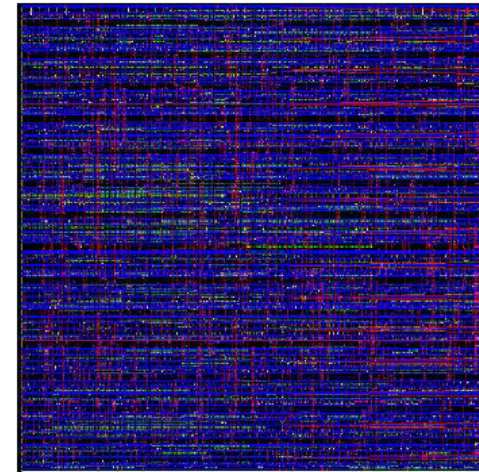
2 input NAND



Sea of NAND gates



Routed NAND gates



Flip-flop from NAND

Performance Parameter	Delay Mapped Ratio (NAND2/ASIC)
Area	1.12
Delay	1.39
Power	1.07

Image and Data Source: A Lithography-friendly Structured ASIC Design Approach

Example 2: Atmel Microcontroller

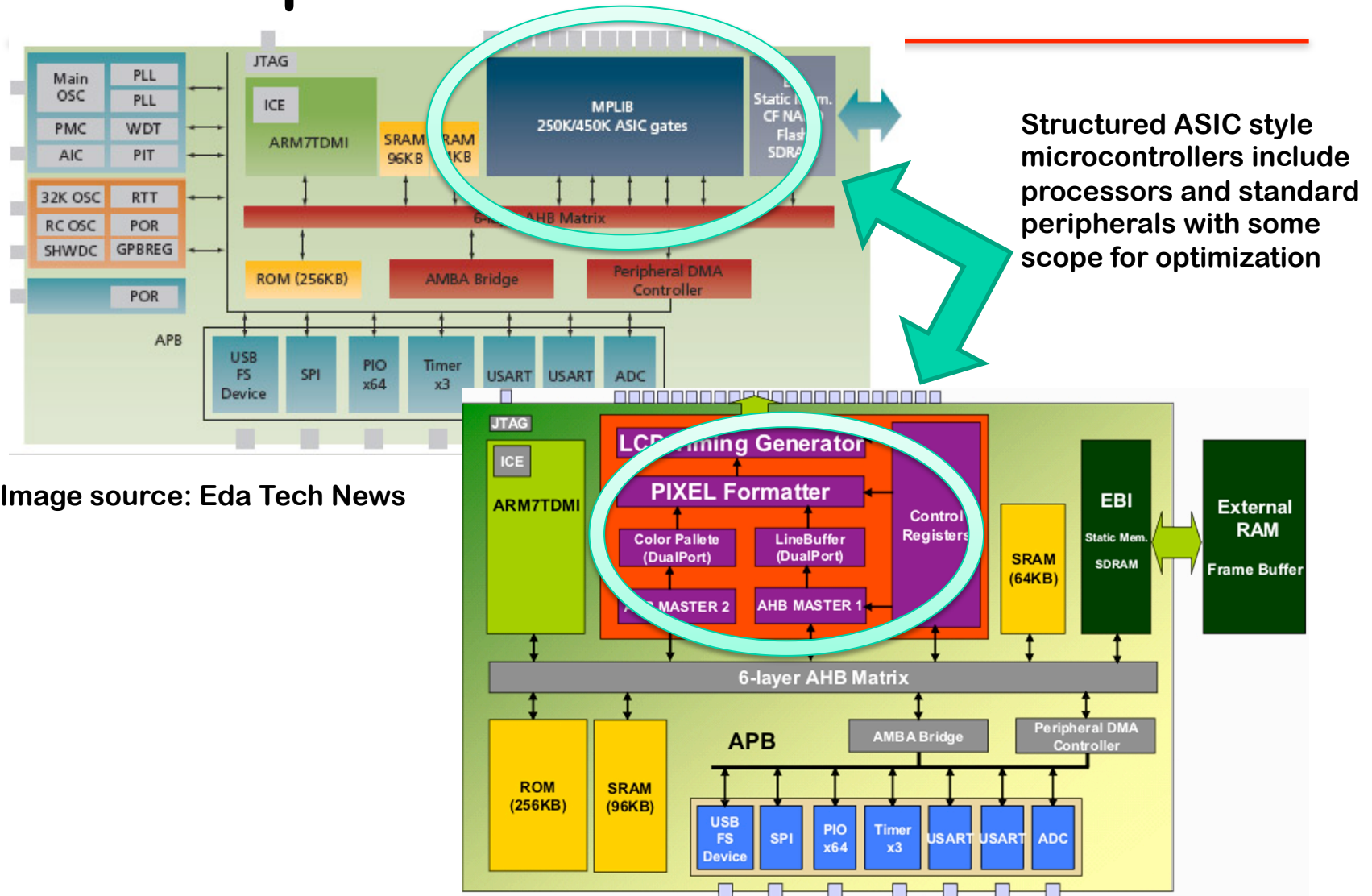


Image source: Eda Tech News

Soft IP store

Power Architecture e200 Z1

Power Architecture e200 Z3

Power Architecture e200 Z6

16-bit Microprocessors

C166S

CR16

8-bit Microprocessors

HCS08

M8051EW+

Test and Debug

Compact JTAG

AMBA Bus Fabric

AHB Backbone

AHB Monitor

AHB-APB Bridge

AMBA AHB System Functions

DMA Controller

RAM Controller

AMBA APB System Functions

Interrupt Controller

Timer and Watchdog

Multifunction Timer

Versatile Timer

Real Time Clock

AMBA Interfaces

USART

I2S Audio Interface

I2C Interface

Advanced Audio Interface

CAN Controller

MICROWIRE/SPI Controller

Gen. Purpose I/O Controller

Automotive

FlexRay Comm. Controller

moment to review the process and if you have any questions relating to the buying experience please email us at core.store@ip-extreme.com

IP Title		Use Price*	Royalty**	Quantity	BUY
Microprocessors					
V1 ColdFire Processor		\$10,000.00	\$0.02		☐
V1 ColdFire Processor for Altera Cyclone III		\$0.00	none		☐
HCS08 Processor		\$10,000.00	\$0.01		☐
AMBA Peripherals					
I2C		\$2,500.00	none		☐
Advanced Audio Interface		\$2,500.00	none		☐
CAN Controller		\$5,000.00	none		☐
General Purpose IO Controller		\$2,500.00	none		☐
I2S Audio Interface		\$2,500.00	none		☐
MICROWIRE/SPI Interface		\$2,000.00	none		☐
Multi-Function Timer		\$3,000.00	none		☐
Real Time Clock		\$3,000.00	none		☐

Choosing design targets to minimize cost

# Mask Sets	Design Style	Explanation	Cost	Product Differentiation
Full mask targets	1. Full-custom, 2. Semi-custom 3. Std-cell (ASIC)	• Complete customization of all mask layers. • Reserved for high-performance, high-volume (microprocessors, analog circuits) • Design libraries can be: • Obtained from external vendor • Full-custom (each team builds one) • Semi-custom (all in-house teams share)	Design cost: Highest Manufacturing cost: Highest	Best: at all levels from fabrication, circuit to high-level design
Metal mask targets	Metal programmable logic (Structured ASIC) Example: Atmel CAP	Wafers with prefabricated array of gates (“sea of universal gates”) and memory/processors that can be customized by connecting wires in layers. Fabs can pre-stock wafers ~ 3 weeks turnaround time.	Design cost: Reasonable Manufacturing cost: Medium	Innovations restricted to functionality (e.g., new USB)
No Masks	Field programmable logic (FPGA) Example: Xilinx, Altera etc.,	“Sea” of lookup tables implement functions Low startup costs, much cheaper and slower than Standard cell designs, for 100K units FPGAs are better.	No fabrication costs! Design cost is same as custom mask options	Usually slower, larger than above two options
No Masks	Soft IP Example (http://www.ip-extreme.com/corestore/)	Provide encrypted intellectual property that can be used by other companies. Initial part and Royalty	Almost like software, need EDA tools	New functionality, better faster etc.

Summary

- **This class**
 - Theory: Design process
- **Next class**
 - SystemVerilog
- **Following theory class**
 - Basic building blocks, control logic etc.,

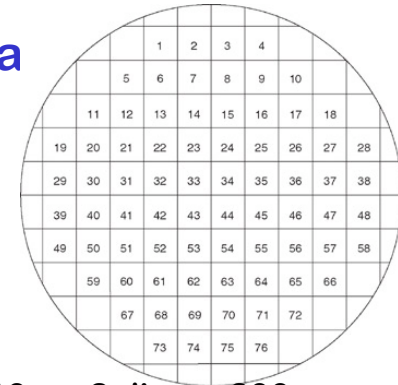
Die Size and Product Cost

- Cost of processing a wafer is independent of die size (to the first order)

- Roughly \$3000 for a 200mm² in 1999 (custom 1a)
- At the same time, 300mm² cost 10X more

$$\text{Die per wafer} = \frac{\pi(\text{wafer diameter}/2)^2}{\text{die area}} - \frac{\pi \times \text{wafer diameter}}{\sqrt{2} \times \text{die area}}$$

$$\text{Die yield} = \text{wafer yield} \times \left(1 + \frac{\text{defects per area} \times \text{die area}}{\alpha}\right)^{-\alpha}$$



310mm² die on 200mm wafer

- Package cost
 - cost = base package cost + cost per pin x # pins
 - Base ~ \$5 for small die, \$20 for large die, and 1 or 2 cents per pin
- Testing cost
 - Cost = test time + test cost per hour
 - Test time = 1-2 minutes, test cost per hour = hundreds per hour

Commodity Die

Assumptions		Calculation	
Die Area	140 mm ²		
Wafer diameter	200 mm	Die per wafer	186
Defect density	0.5/cm ²		
Process complexity	4		
Wafer yield	95%	Die yield	50%
Processed Wafer Cost	\$3000	Die cost	\$33
Base package cost	\$10		
Cost per pin	\$0.01		
Number of pins	500	Package cost	\$15
Test time	30s		
Test cost per hour	\$400/hour	Test cost	\$3
Test yield	95%	Processor cost	\$54

Data from: Microprocessor Design by Grant McFarland

Server Die

Assumptions		Calculation	
Die Area	310 mm ²		
Wafer diameter	200 mm	Die per wafer	76
Defect density	0.5/cm ²		
Process complexity	4		
Wafer yield	95%	Die yield	25%
Processed Wafer Cost	\$3000	Die cost	\$158
Base package cost	\$15		
Cost per pin	\$0.01		
Number of pins	1000	Package cost	\$25
Test time	45s		
Test cost per hour	\$400/hour	Test cost	\$5
Test yield	95%	Processor cost	\$198

Data from: Microprocessor Design by Grant McFarland

Comparison

	Commodity	Server
Die Cost	64%	84%
Package and Assembly	29%	13%
Test	7%	3%

Observations:

Know when to optimize for area, and remember each design decision affects cost!

Data from: Microprocessor Design by Grant McFarland