Chapter 1

Computer Abstractions and Technology

UNIVERSITY OF SOUTH CAROLINA

What is Computer Architecture?

- The design of computer systems

- Goal: improve "performance":
  - Run programs faster
    - Execution time: e.g. less waiting time, more simultaneous tasks
    - Throughput: e.g. higher framerate, faster downloads
  - Use less power, last longer on battery power
  - Generate less or more uniformally-distributed heat
  - Handle more secure encryption standards
  - Software defined networking at higher line speeds
  - More scalable
  - Less expensive

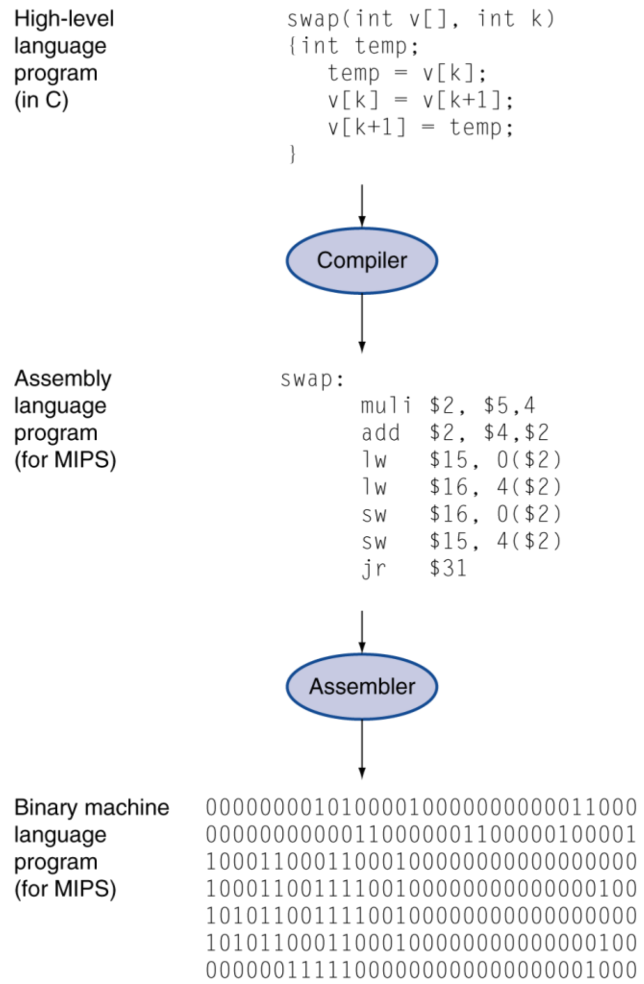
Computer Architecture

- Instruction Set Architecture (“architecture”)
  - The native programming language of a processor
    - Assembly language
    - Machine language
  - Openly published to users, licensed for chip makers

- Microarchitecture
  - The internal organization of a processor
  - Executes programs
  - Trade secret

Levels of Program Code

- High-level language
  - Level of abstraction closer to problem domain
  - Provides for productivity and portability
- Assembly language
  - Textual representation of instructions
- Hardware representation
  - Binary digits (bits)
  - Encoded instructions and data
- Instruction Set Architecture (ISA):
  - Assembly code / machine code “language”
- Microarchitecture:
  - ISA implementation



Abstraction

- Abstration used to manage complexity of design
  - Hide details that are not important

Program code

↓

Machine Instructions

↓

Datapaths

↓

Logic gates

↓

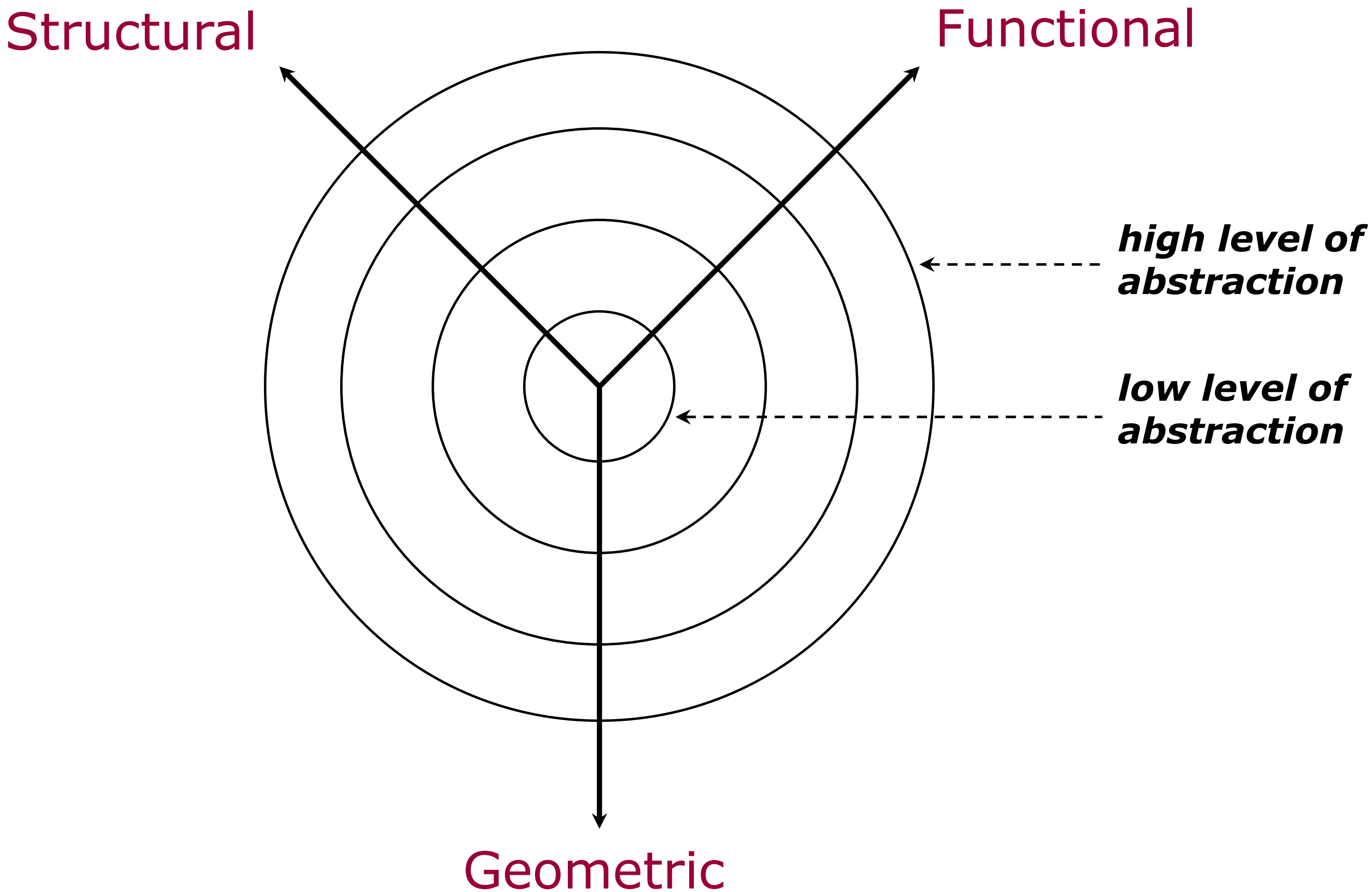
Devices (Transistors)

Why Study Computer Architecture

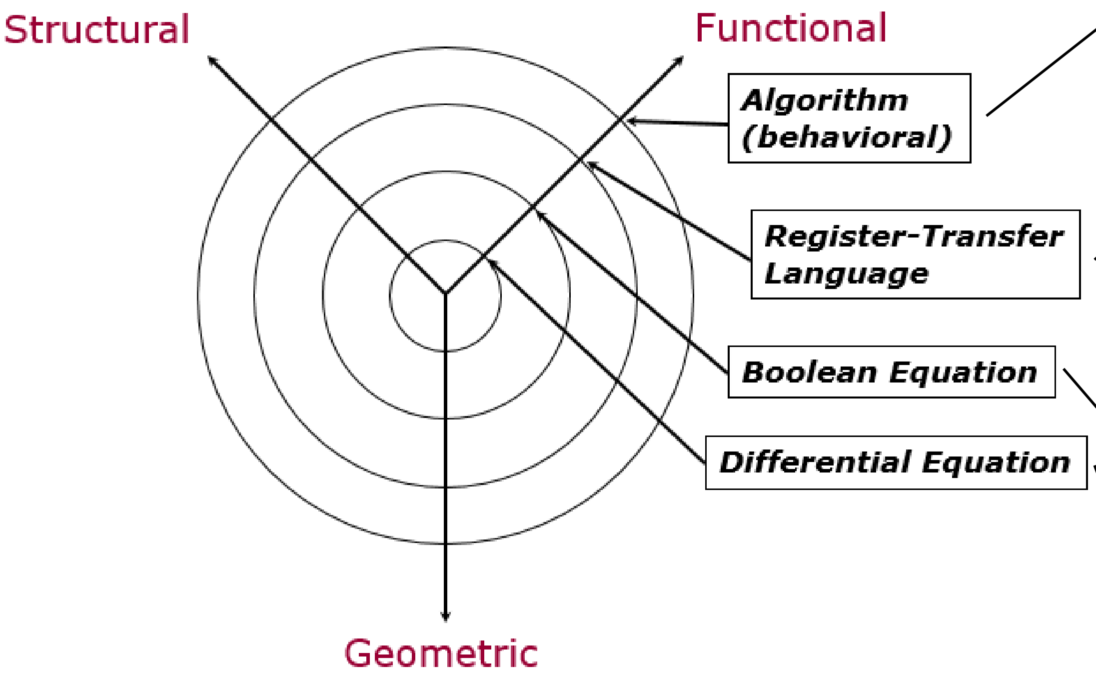
- Compiling "machine agnostic" code:
  - Generally achieve ~1-20% of peak theoretical performance
  - Performance tuned code must be explicitly written for the underlying architecture
  - Especially for **embedded** and **special purpose** processors
  - Understanding computer architecture allows for customization:
    - Multicore
    - More efficient use of registers, instructions

- Device drivers must directly interface with peripherals
  - Uses CPU-specific, bit-level features to communicate
  - E.g. memory-mapped I/O, interrupts, DMA, double buffering, bit fields in status/control registers, memory management, virtual memory

Domains and Levels of Modeling



Functional Abstraction




```
For i=0 to 10
    C = C + A[i]
```

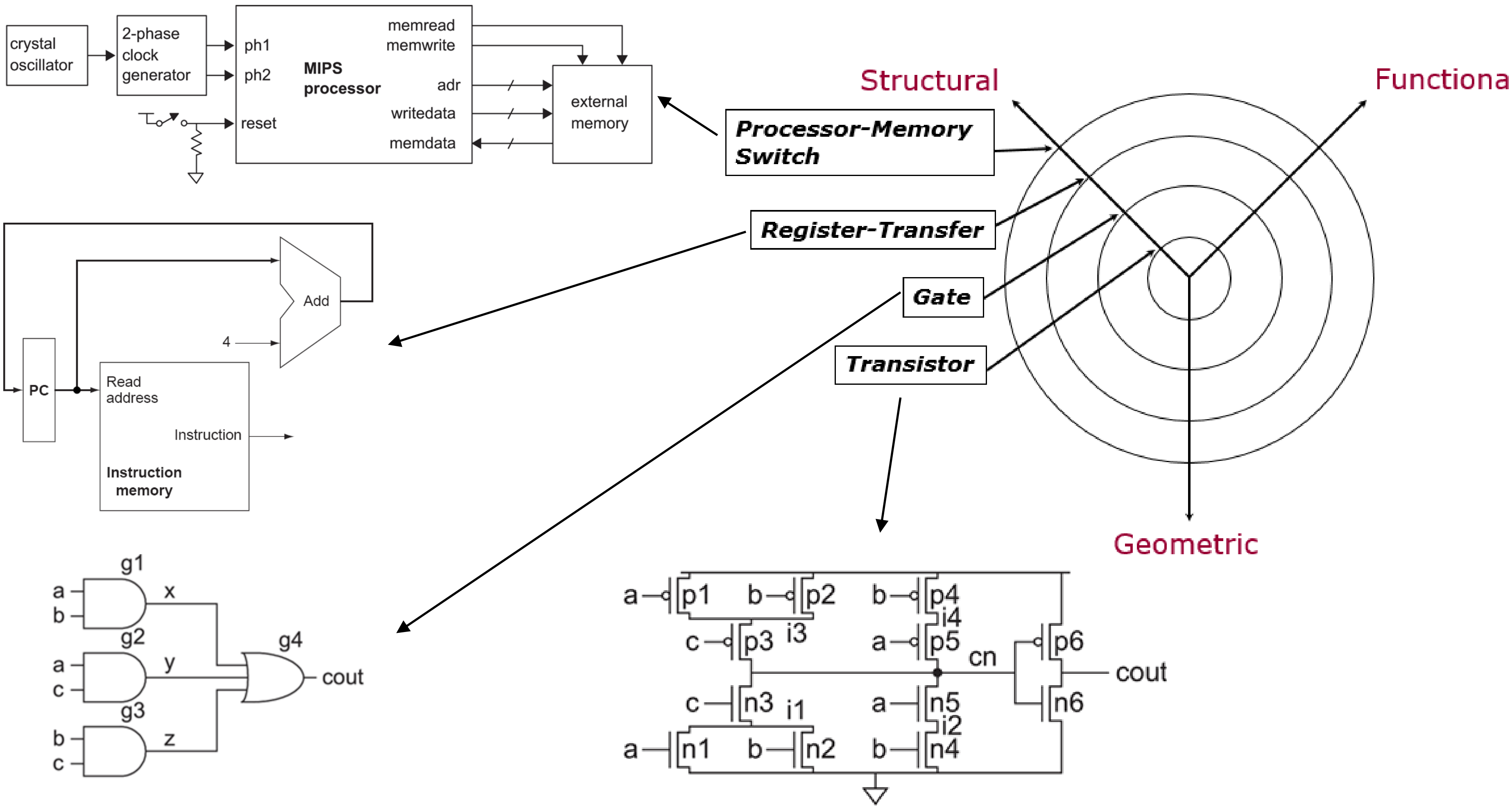
```
MAR <= PC, memory_read <= 1
PC <= PC + 1
wait until ready = 1
IR <= memory_data
memory_read <= 0
```

$$C = XY + (X \oplus Y)Z$$

$$S = X \oplus Y \oplus Z$$

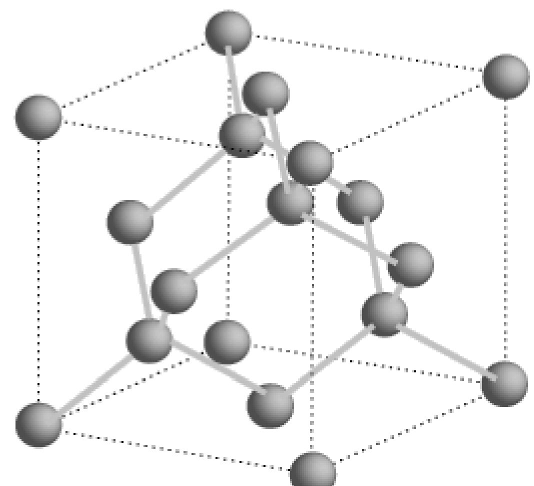
$$\frac{\partial}{\partial x} V(x,t) = -L \frac{\partial}{\partial t} I(x,t)$$

Structural Abstraction



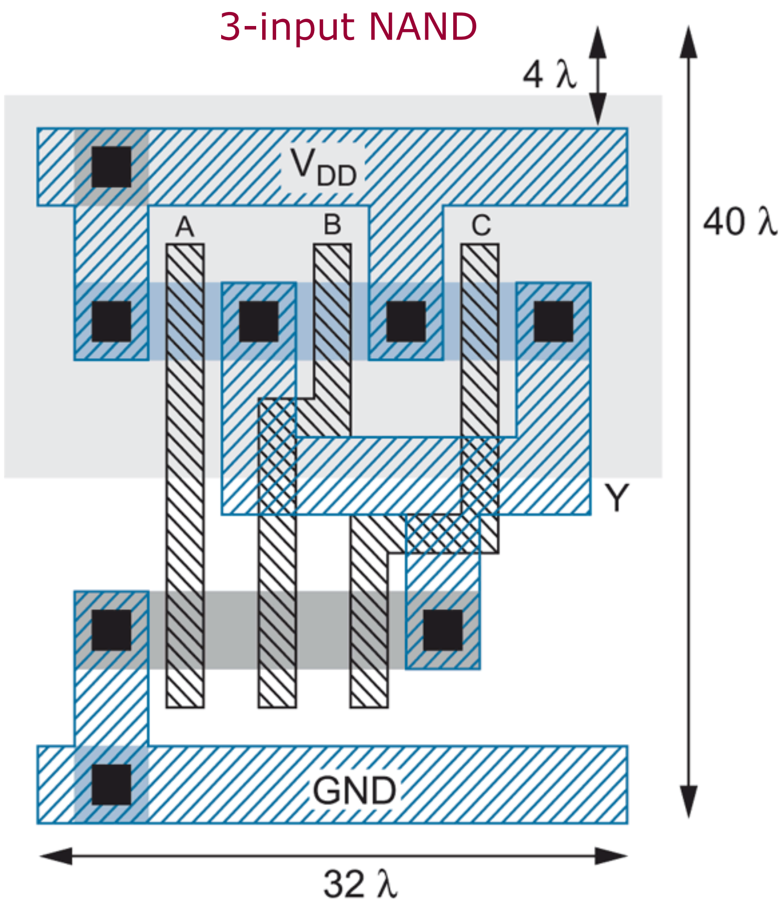
Semiconductors

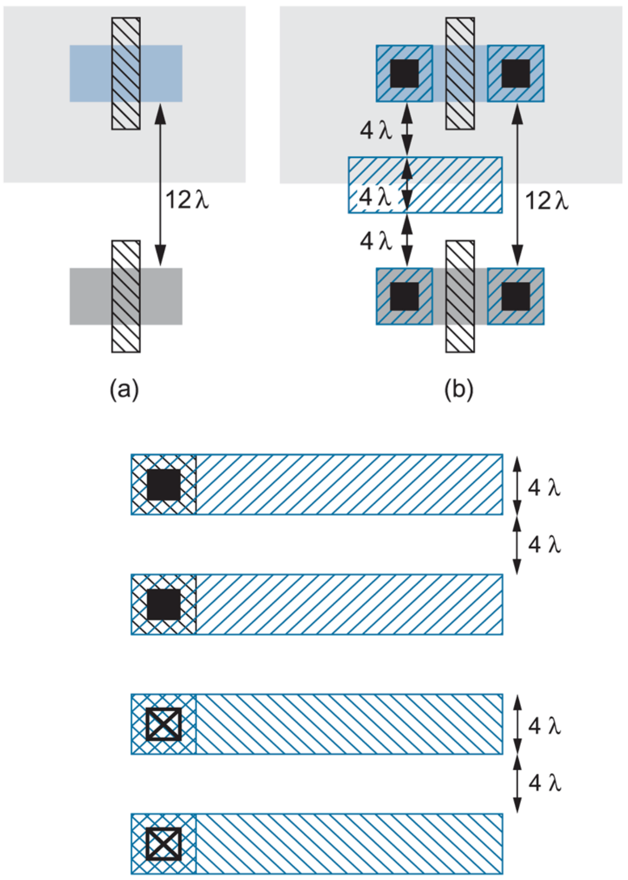
- Silicon is a group IV element
- Forms covalent bonds with four neighbor atoms (3D cubic crystal lattice)
- Si is a poor conductor, but *conduction* characteristics may be altered
- Add impurities/dopants replaces silicon atom in lattice
  - Adds two different types of *charge carriers*



Spacing = .543 nm

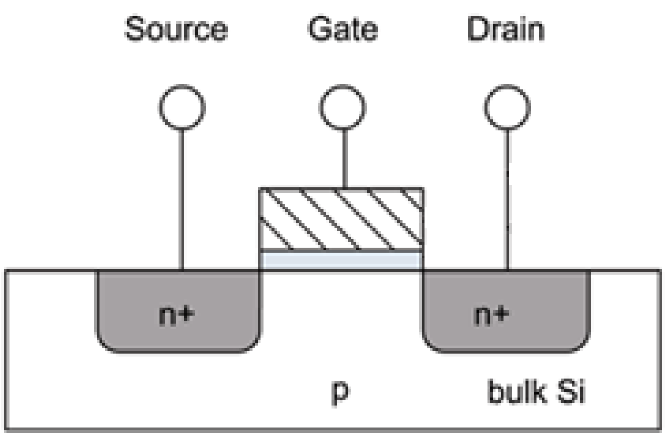
Layout


3-input NAND
4 λ
40 λ
32 λ
V_{DD}
A
B
C
Y
GND


12 λ
4 λ
4 λ
4 λ
12 λ
(a)
(b)
4 λ
4 λ
4 λ
4 λ

Feature Size

- Shrink minimum feature size…
  - Smaller L decreases carrier time and increases current
  - Therefore, W may also be reduced for fixed current
  - C_g, C_s, and C_d are reduced
  - Transistor switches faster (~*linear* relationship)



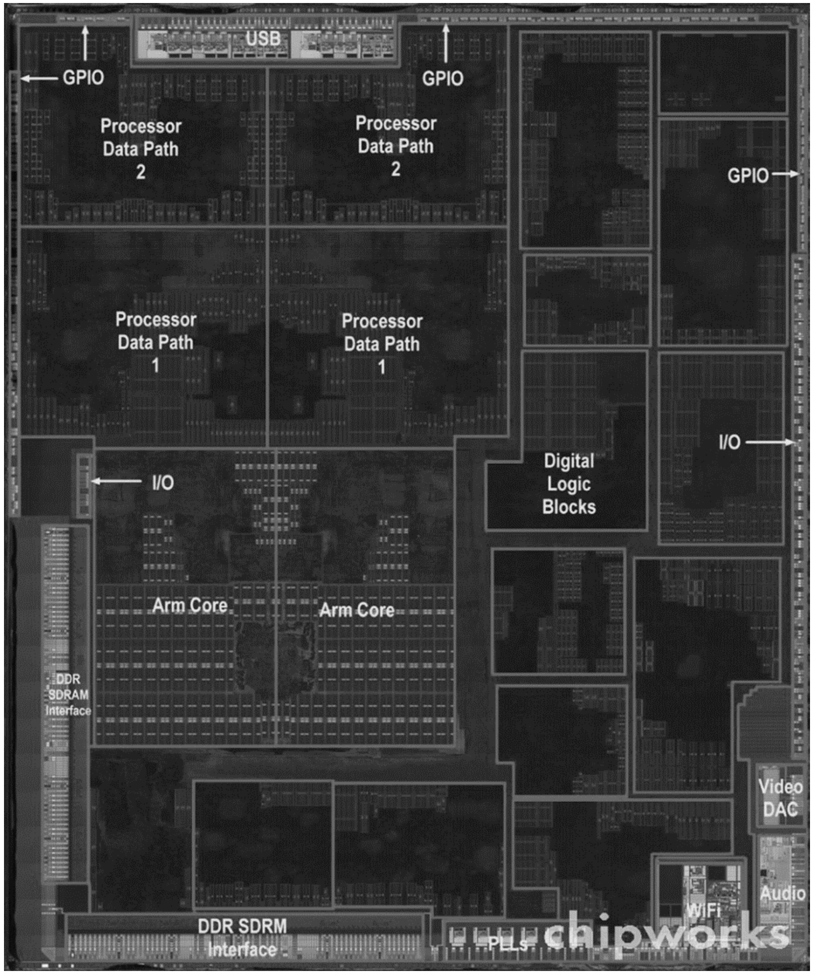
Minimum Feature Size

Year	Processor	Performance	Transistor Size	Transistors
1982	i286	6 - 25 MHz	1.5 μm	~134,000
1986	i386	16 – 40 MHz	1 μm	~270,000
1989	i486	16 - 133 MHz	.8 μm	~1 million
1993	Pentium	60 - 300 MHz	.6 μm	~3 million
1995	Pentium Pro	150 - 200 MHz	.5 μm	~4 million
1997	Pentium II	233 - 450 MHz	.35 μm	~5 million
1999	Pentium III	450 – 1400 MHz	.25 μm	~10 million
2000	Pentium 4	1.3 – 3.8 GHz	.18 μm	~50 million
2005	Pentium D	2 threads/package	.09 μm	~200 million
2006	Core 2	2 threads/die	.065 μm	~300 million
2008	"Nehalem"	8 threads/die	.045 μm	~800 million
2009	"Westmere"	8 threads/die	.045 μm	~1 billion
2011	"Sandy Bridge"	12 threads/die	.032 μm	~1.2 billion
2013	"Ivy Bridge"	16 threads/die	.022 μm	~1.4 billion

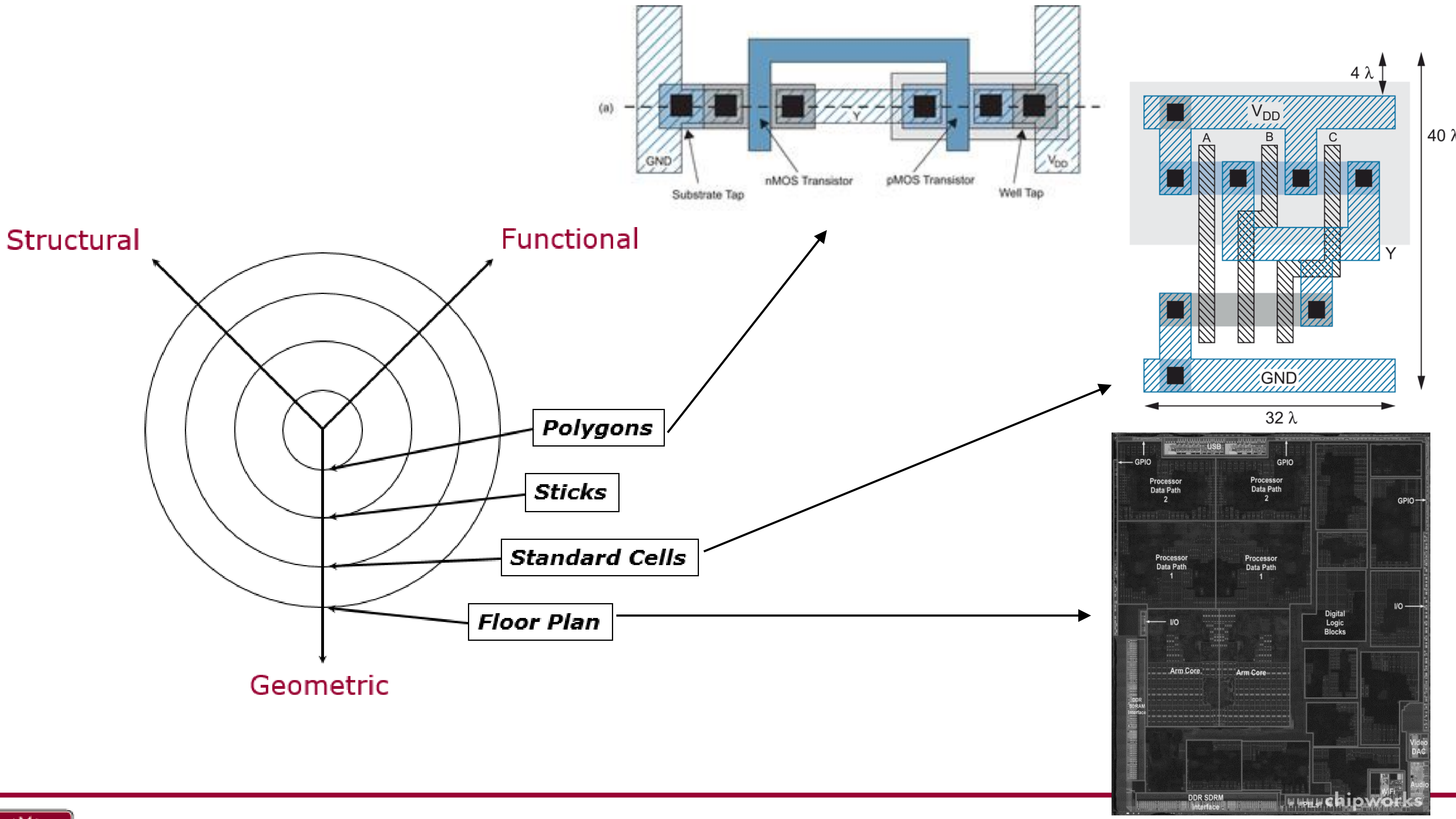
Year	Processor	Speed	Transistor Size	Transistors
2008	NVIDIA Tesla (GT200)	240 threads/die	.065 μm	1.4 billion
2010	NVIDIA Fermi (GF110)	512 threads/die	.040 μm	3.0 billion
2012	NVNDIA Kepler (GK104)	1536 threads/die	.028 μm	3.5 billion

Inside the Processor

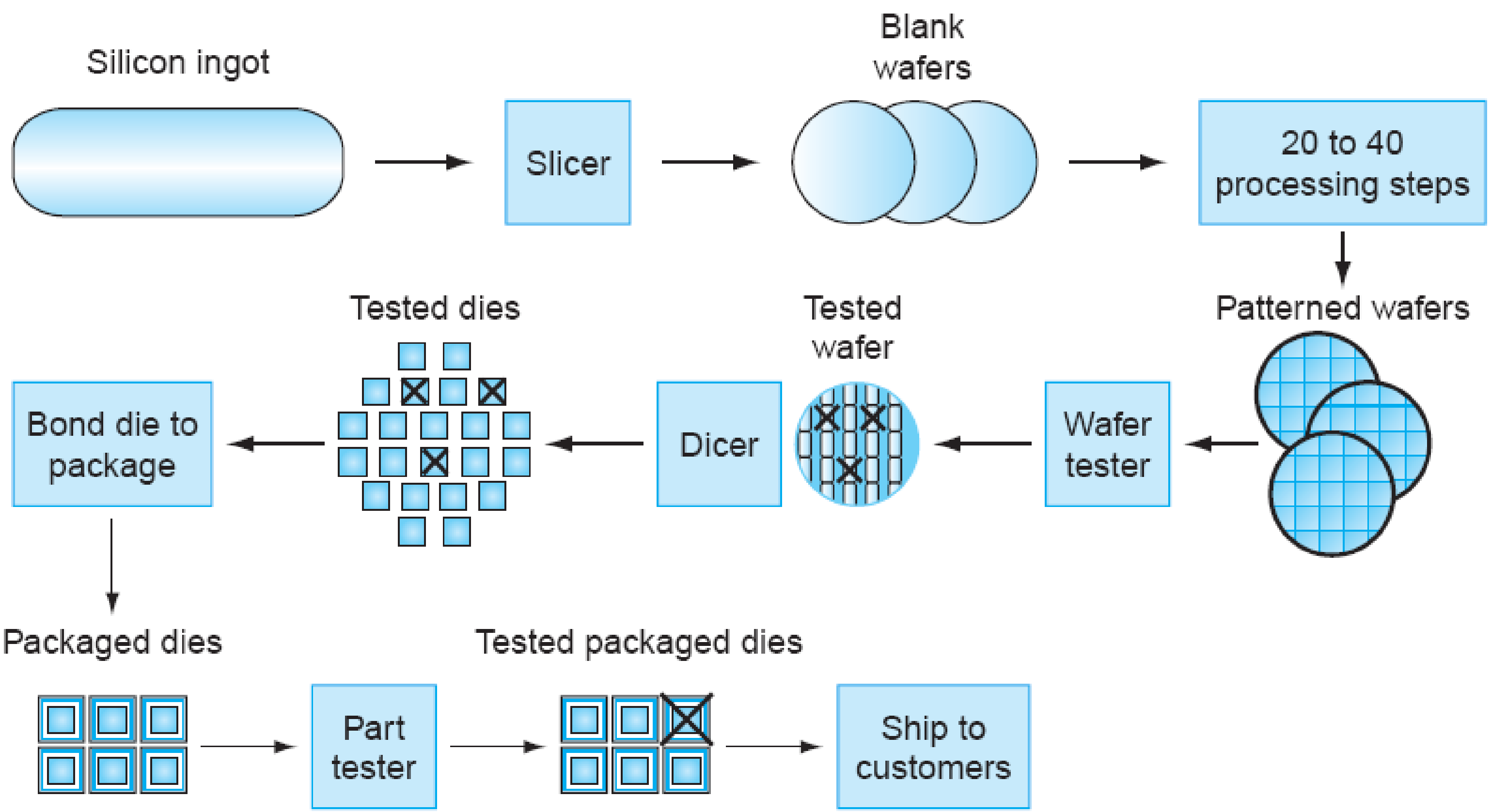
- Apple A5



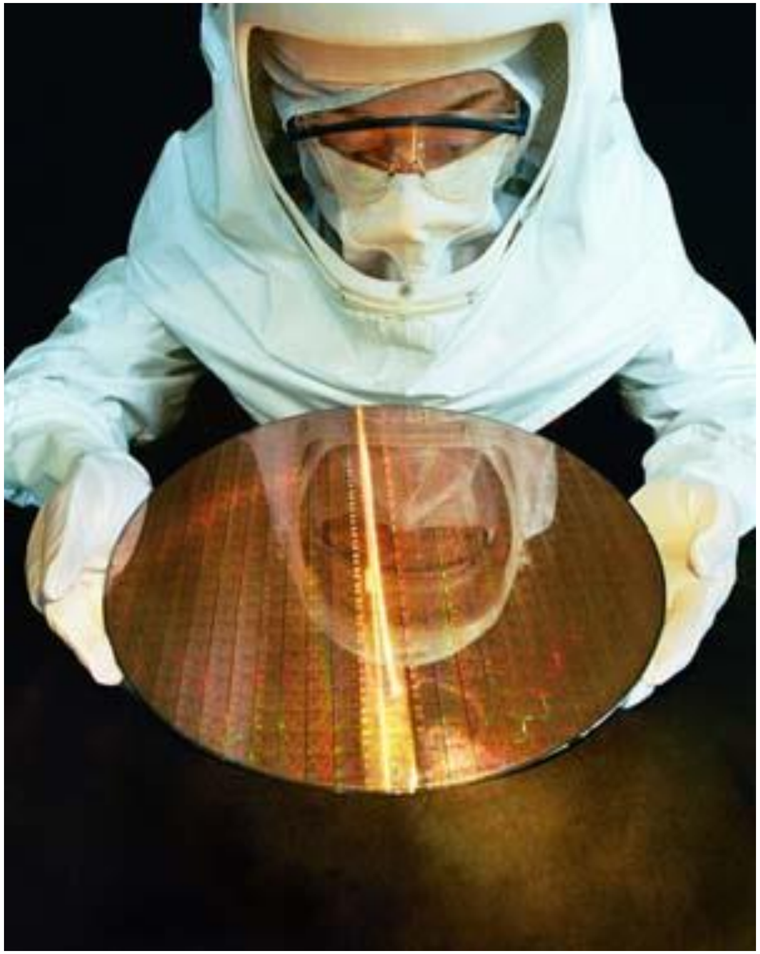
Geometric Abstraction

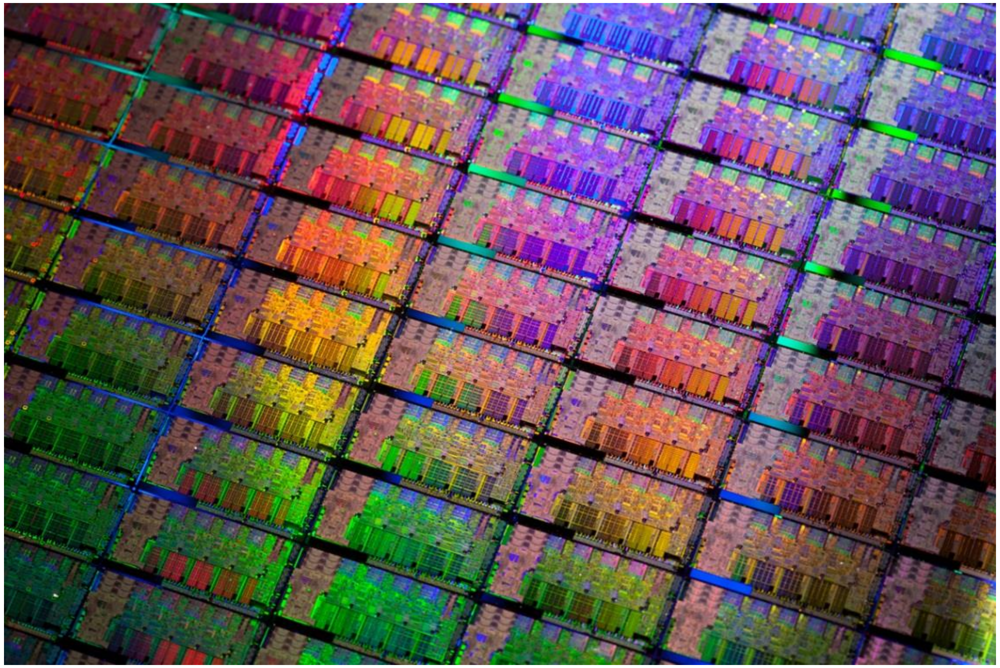


IC Fabrication

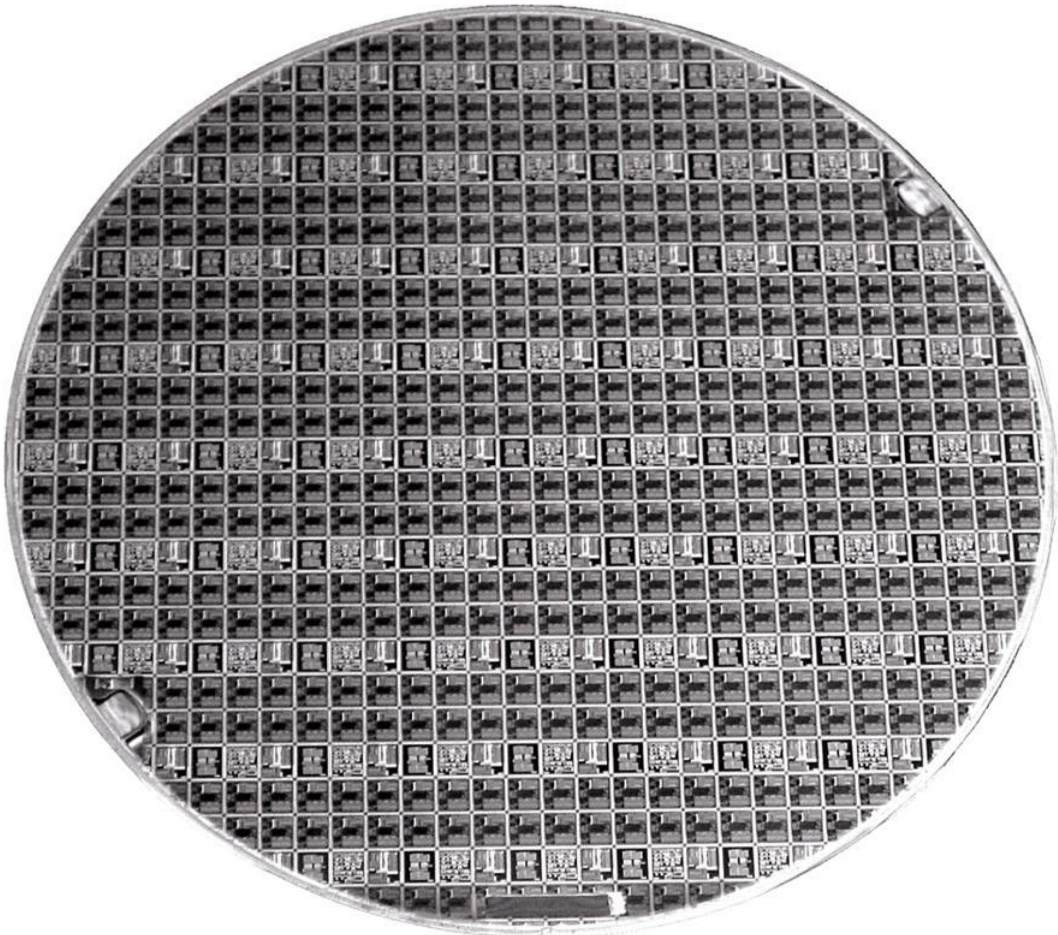


Si Wafer

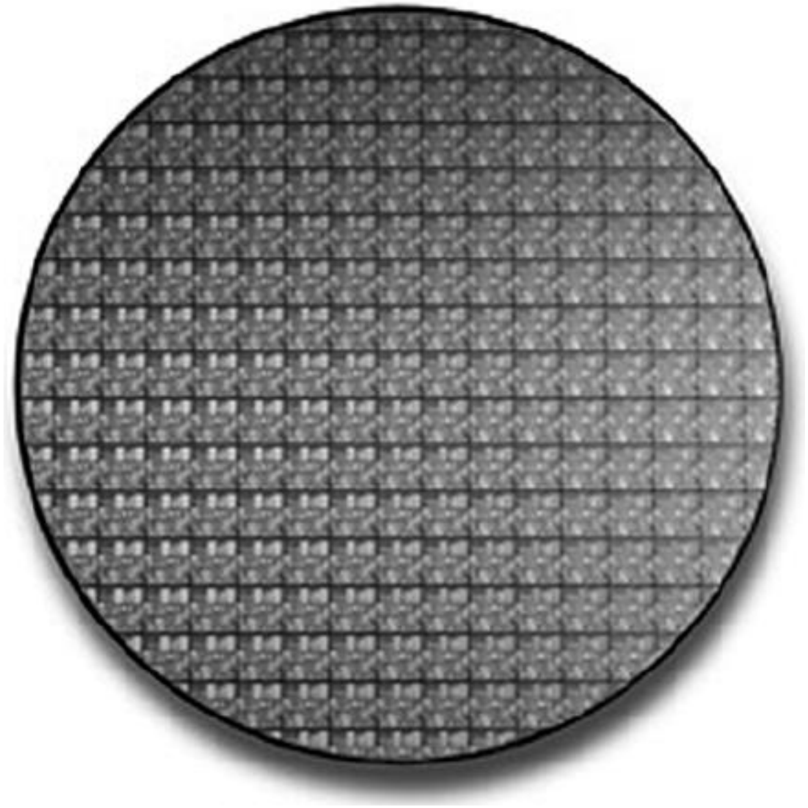




8″ Wafer

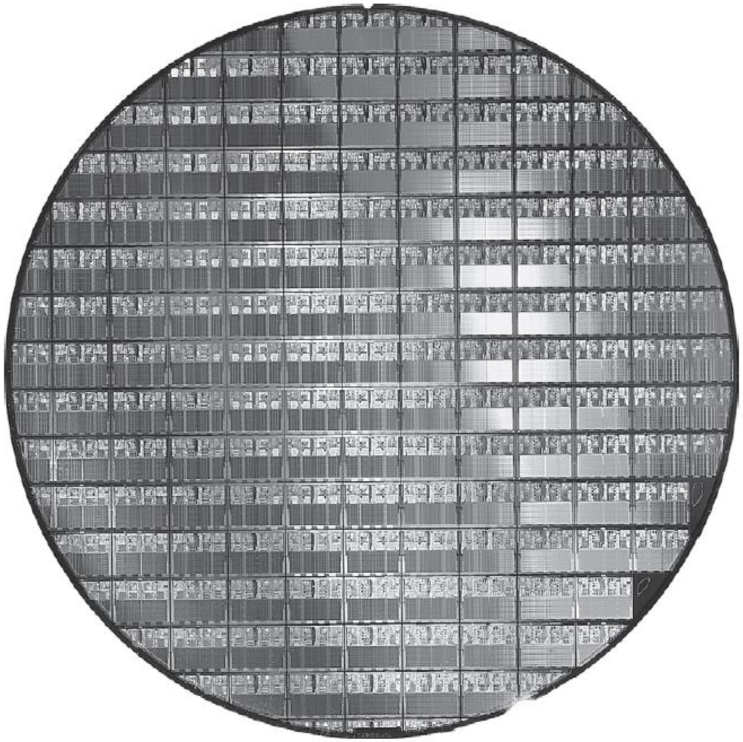


8″ Wafer



- 8 inch (200 mm) wafer containing Pentium 4 processors
  - 165 dies, die area = 250 mm^2, 55 million transistors, .18μm

Intel Core i7 Wafer



- 300mm wafer, 280 chips, 32nm technology
- Each chip is 20.7 x 10.5 mm

Speedup / Relative Performance

- Define Performance = 1/Execution Time
- "X is n time faster than Y"

$$\text{Performance}_X / \text{Performance}_Y = \text{Execution time}_Y / \text{Execution time}_X = n$$

- Example: time taken to run a program
  - 10s on A, 15s on B
  - Execution Time_B / Execution Time_A = 15s / 10s = 1.5
  - So A is 1.5 times faster than B

Integrated Circuit Cost

$$\text{Cost per die} = \frac{\text{Cost per wafer}}{\text{Dies per wafer} \times \text{Yield}}$$

$$\text{Dies per wafer} \approx \text{Wafer area} / \text{Die area}$$

$$\text{Yield} = \frac{1}{(1 + (\text{Defects per area} \times \text{Die area}/2))^2}$$

- Nonlinear relation to area and defect rate
  - Wafer cost and area are fixed
  - Defect rate determined by manufacturing process
  - Die area determined by architecture and circuit design

Example

- Assume C defects per area and a die area of D. Calculate the improvement in yield if the number of defects is reduced by 1.5.

$$\frac{yield_{new}}{yield_{orig}} = \frac{\frac{1}{\left(1+\frac{C}{1.5}\frac{D}{2}\right)^2}}{\frac{1}{\left(1+C\frac{D}{2}\right)^2}} =$$

$$\frac{\left(1+C\frac{D}{2}\right)^2}{\left(1+\frac{C}{1.5}\frac{D}{2}\right)^2}$$

$$= \frac{1+CD+\frac{C^2D^2}{4}}{1+\frac{CD}{1.5}+\frac{C^2D^2}{9}}$$

$$\text{Cost per die} = \frac{\text{Cost per wafer}}{\text{Dies per wafer} \times \text{Yield}}$$

$$\text{Dies per wafer} \approx \text{Wafer area/Die area}$$

$$\text{Yield} = \frac{1}{(1+(\text{Defects per area} \times \text{Die area/2}))^2}$$

Example

$$\text{Cost per die} = \frac{\text{Cost per wafer}}{\text{Dies per wafer} \times \text{Yield}}$$

$$\text{Dies per wafer} \approx \text{Wafer area/Die area}$$

$$\text{Yield} = \frac{1}{(1 + (\text{Defects per area} \times \text{Die area/2}))^2}$$

- Assume a 20 cm diameter wafer has a cost of 15, contains 100 dies, and has 0.031 defects/cm^2.

1. If the number of dies per wafer is increased by 10% and the defects per area unit increases by 15%, find the die area and yield.

die area$_{20cm}$ = wafer area / dies per wafer = pi*10^2 / (100*1.1) = 2.86 cm^2

yield$_{20cm}$ = 1/(1+(0.03*1.15* 2.86/2))^2 = 0.9082

2. Assume a fabrication process improves the yield from 0.92 to 0.95. Find the defects per area unit for each version of the technology given a die area of 200 mm^2.

defects per area$_{0.92}$ = (1-y^.5)/(y^.5*die_area/2) =
(1-0.92^.5)/(0.92^.5*2/2) = 0.043 defects/cm^2

defects per area$_{0.95}$ = (1-y^.5)/(y^.5*die_area/2) =
(1-0.95^.5)/(0.95^.5*2/2) = 0.026 defects/cm^2

Response Time and Throughput

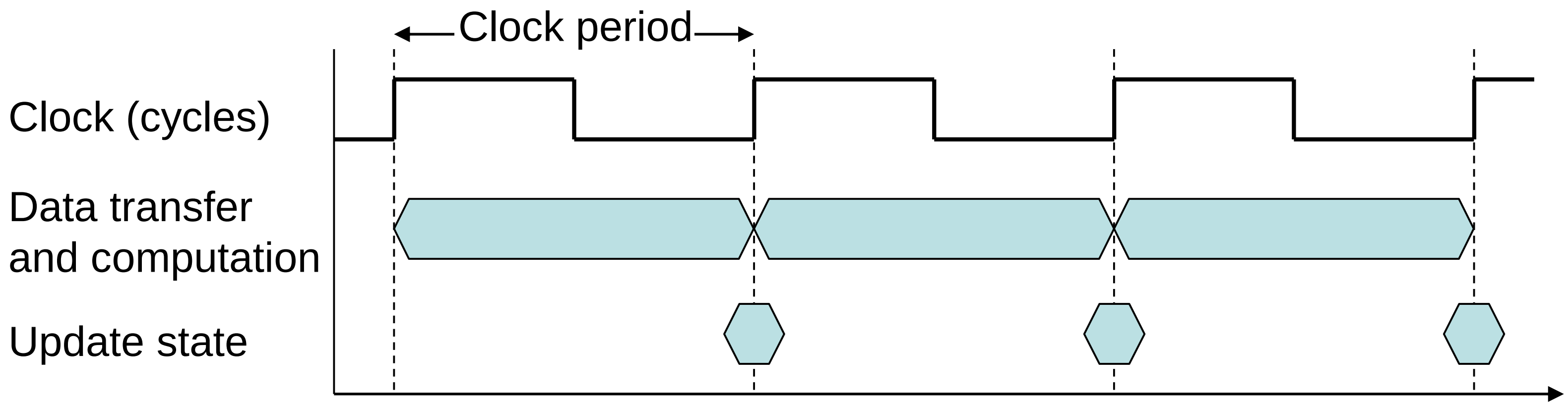
- Response time
  - How long it takes to do a task
- Throughput
  - Total work done per unit time
    - e.g., tasks/transactions/… per hour
- How are response time and throughput affected by
  - Replacing the processor with a faster version?
  - Adding more processors?
- We'll focus on response time for now…

Measuring Execution Time

- Elapsed time
  - Total response time, including all aspects
    - Processing, I/O, OS overhead, idle time
  - Determines system performance
- CPU time
  - Time spent processing a given job
    - Discounts I/O time, other jobs' shares
  - Comprises user CPU time and system CPU time
  - Different programs are affected differently by CPU and system performance

CPU Clocking

- Operation of digital hardware governed by a constant-rate clock



  - Clock period: duration of a clock cycle
    - e.g., 250ps = 0.25ns = 250×10^{-12} s
  - Clock frequency (rate): cycles per second
    - e.g., 4.0GHz = 4000MHz = 4.0×10^{9} Hz

CPU Time

$$\text{CPU Time} = \text{CPU Clock Cycles} \times \text{Clock Cycle Time}$$

$$= \frac{\text{CPU Clock Cycles}}{\text{Clock Rate}}$$

- Performance improved by
  - Reducing number of clock cycles
  - Increasing clock rate
  - Hardware designer must often trade off clock rate against cycle count

CPU Time Example

- Computer A: 2GHz clock, 10s CPU time
- Designing Computer B
  - Aim for 6s CPU time
  - Can do faster clock, but causes 1.2 × clock cycles
- How fast must Computer B clock be?

$$\text{Clock Rate}_B = \frac{\text{Clock Cycles}_B}{\text{CPU Time}_B} = \frac{1.2 \times \text{Clock Cycles}_A}{6s}$$

$$\text{Clock Cycles}_A = \text{CPU Time}_A \times \text{Clock Rate}_A$$

$$= 10s \times 2\text{GHz} = 20 \times 10^9$$

$$\text{Clock Rate}_B = \frac{1.2 \times 20 \times 10^9}{6s} = \frac{24 \times 10^9}{6s} = 4\text{GHz}$$

Instruction Count and CPI

$$\text{Clock Cycles} = \text{Instruction Count} \times \text{Cycles per Instruction}$$

$$\text{CPU Time} = \text{Instruction Count} \times \text{CPI} \times \text{Clock Cycle Time}$$

$$= \frac{\text{Instruction Count} \times \text{CPI}}{\text{Clock Rate}}$$

- Instruction Count for a program
  - Determined by program, ISA and compiler
- Average cycles per instruction
  - Determined by CPU hardware
  - If different instructions have different CPI
    - Average CPI affected by instruction mix

CPI Example

- Computer A: Cycle Time = 250ps, CPI = 2.0
- Computer B: Cycle Time = 500ps, CPI = 1.2
- Same ISA
- Which is faster, and by how much?

$$\text{CPU Time}_A = \text{Instruction Count} \times \text{CPI}_A \times \text{Cycle Time}_A$$

$$= I \times 2.0 \times 250\text{ps} = I \times 500\text{ps}$$

A is faster…

$$\text{CPU Time}_B = \text{Instruction Count} \times \text{CPI}_B \times \text{Cycle Time}_B$$

$$= I \times 1.2 \times 500\text{ps} = I \times 600\text{ps}$$

$$\frac{\text{CPU Time}_B}{\text{CPU Time}_A} = \frac{I \times 600\text{ps}}{I \times 500\text{ps}} = 1.2$$

…by this much

CPI in More Detail

- If different instruction classes take different numbers of cycles

$$\text{Clock Cycles} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{Instruction Count}_i)$$

- Weighted average CPI

$$\text{CPI} = \frac{\text{Clock Cycles}}{\text{Instruction Count}} = \sum_{i=1}^{n} \left(\text{CPI}_i \times \underbrace{\frac{\text{Instruction Count}_i}{\text{Instruction Count}}}_{\text{Relative frequency}} \right)$$

CPI Example

- Alternative compiled code sequences using instructions in classes A, B, C

Class	A	B	C
CPI for class	1	2	3
IC in sequence 1	2	1	2
IC in sequence 2	4	1	1

- Sequence 1: IC = 5
  - Clock Cycles
    = 2×1 + 1×2 + 2×3
    = 10
  - Avg. CPI = 10/5 = 2.0
- Sequence 2: IC = 6
  - Clock Cycles
    = 4×1 + 1×2 + 1×3
    = 9
  - Avg. CPI = 9/6 = 1.5

Performance Summary

$$\text{CPU Time} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}}$$

- Performance depends on
  - Algorithm: affects IC, possibly CPI
  - Programming language: affects IC, CPI
  - Compiler: affects IC, CPI
  - Instruction set architecture: affects IC, CPI, T_c

Example

- Suppose one machine, A, executes a program with an average CPI of 2.1
- Suppose another machine, B (with the same instruction set and an enhanced compiler), executes the same program with 25% less instructions and with a CPI of 1.8 at 800MHz

In order for the two machines to have the same performance, what does the clock rate of the first machine (machine A) need to be?

$$\frac{I_A CPI_A}{R_A} = \frac{I_B CPI_B}{R_B}$$

$$\frac{I_A \cdot 2.1}{R_A} = \frac{0.75 I_A \cdot 1.8}{800 \cdot 10^6}$$

Example

- Suppose a program has the following instruction classes, CPIs, and mixtures:

Instruction type	CPI	ratio
A	1.4	55%
B	2.4	15%
C	2	30%

Your engineers give you the following options:

Option A: Reduce the CPI of instruction type A to 1.1

Option B: Reduce the CPI of instruction type B to 1.2

Which option would you choose and why?

$$CPI_A = .55(1.1) + .15(2.4) + .30(2) = 1.565$$

$$CPI_B = .55(1.4) + .15(1.2) + .30(2) = 1.550$$

Pitfall: MIPS as a Performance Metric

- MIPS: Millions of Instructions Per Second
  - Doesn't account for
    - Differences in ISAs between computers
    - Differences in complexity between instructions

$$\begin{aligned}\text{MIPS} &= \frac{\text{Instruction count}}{\text{Execution time} \times 10^6} \\ &= \frac{\text{Instruction count}}{\dfrac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}} \times 10^6} = \frac{\text{Clock rate}}{\text{CPI} \times 10^6}\end{aligned}$$

- CPI varies between programs on a given CPU

Example

- Consider two different implementations, M1 and M2, of the same instruction set. There are three classes of instructions (A, B, and C) in the instruction set. M1 has a clock rate of 800 MHz and M2 has a clock rate of 2 GHz. The average number of cycles for each instruction class and their frequencies (for a typical program) are as follows:

Instruction class	Machine M1 CPI	Frequency	Machine M2 CPI	Frequency
A	1	50%	2	60%
B	2	20%	3	30%
C	4	30%	4	10%

Calculate the average CPI for each machine, M1, and M2.

M1 => .5(1) + .2(2) + .3(4) = 2.1
M2 => .6(2) + .3(3) + .1(4) = 2.5

Calculate the average MIPS ratings for each machine, M1 and M2.

Hint: MIPS = (clock rate / CPI) / 10^6.

M1 => 800 * 2.1 = 1680
M2 => 2000 * 2.5 = 5000

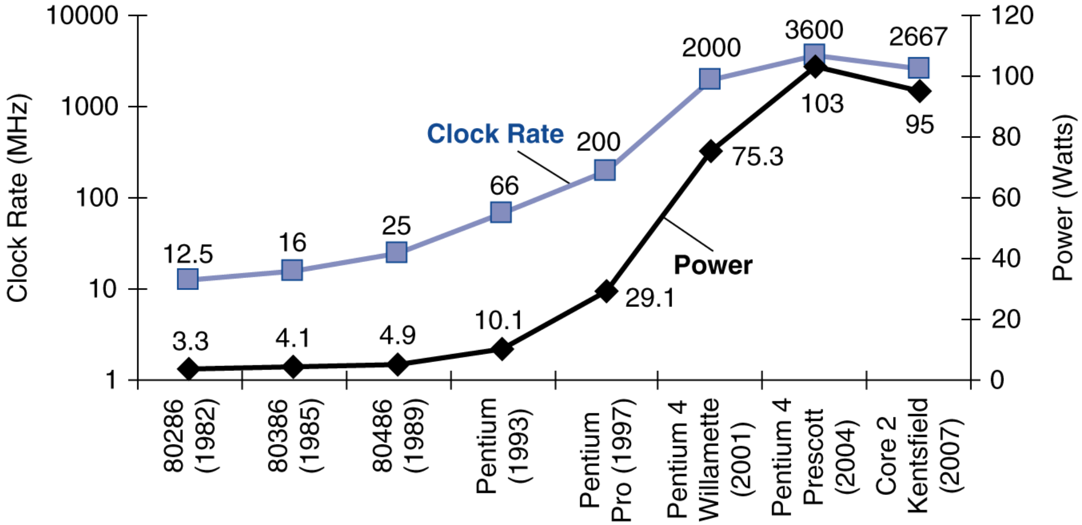
Example (Con't)

- How many less instructions would M1 need to execut to match the speed of M2?

- M1 => 800 * 2.1 = 1680
- M2 => 2000 * 2.5 = 5000

5000/1680

Power Trends



- In CMOS IC technology

$$\text{Power} = \text{Capacitive load} \times \text{Voltage}^2 \times \text{Frequency}$$

×30

5V → 1V

×1000

Reducing Power

- Suppose a new CPU has
  - 85% of capacitive load of old CPU
  - 15% voltage and 15% frequency reduction

$$\frac{P_{new}}{P_{old}} = \frac{C_{old} \times 0.85 \times (V_{old} \times 0.85)^2 \times F_{old} \times 0.85}{C_{old} \times V_{old}{}^2 \times F_{old}} = 0.85^4 = 0.52$$

- The power wall
  - We can't reduce voltage further
  - We can't remove more heat
- How else can we improve performance?

Example

- Assume:

Processor	Clock	Voltage	Dynamic P	Static P
Pentium 4	3.6 GHz	1.25 V	90 W	10 W
Core i5	3.4 GHz	0.9 V	40 W	30 W

Calculate capacitive load of each processor.

If total power is to be reduced by 10%, how much should the voltage be reduced?

Example

- For given processor, assume we reduce the voltage by 10% and increase the frequency by 5%. What is the improvement to dynamic power consumption?

$$\frac{power_{orig}}{power_{new}} = \frac{CV^2F}{C(.9V)^2(1.05)F} = \frac{V^2}{(.9V)^2(1.05)}$$

$$= \frac{V^2}{.81V^2(1.05)} = \frac{1}{.81(1.05)} = 1.18$$

Fallacy: Low Power at Idle

- i7 power benchmark
  - At 100% load: 258W
  - At 50% load: 170W (66%)
  - At 10% load: 121W (47%)
- Google data center
  - Mostly operates at 10% – 50% load
  - At 100% load less than 1% of the time
- Consider designing processors to make power proportional to load

SPEC Power Benchmark

- Power consumption of server at different workload levels
  - Performance: ssj_ops/sec
    - ssj_ops = server side Java operations per second
  - Power: Watts (Joules/sec)

$$\text{Overall ssj\_ops per Watt} = \left(\sum_{i=0}^{10} \text{ssj\_ops}_i \right) \Big/ \left(\sum_{i=0}^{10} \text{power}_i \right)$$

SPECpower_ssj2008 for Xeon X5650

Target Load %	Performance (ssj_ops)	Average Power (Watts)
100%	865,618	258
90%	786,688	242
80%	698,051	224
70%	607,826	204
60%	521,391	185
50%	436,757	170
40%	345,919	157
30%	262,071	146
20%	176,061	135
10%	86,784	121
0%	0	80
Overall Sum	4,787,166	1,922
Σssj_ops/Σpower =		2,490

SPEC CPU Benchmark

- Programs used to measure performance
  - Supposedly typical of actual workload
- Standard Performance Evaluation Corp (SPEC)
  - Develops benchmarks for CPU, I/O, Web, …
- SPEC CPU2006
  - Elapsed time to execute a selection of programs
    - Negligible I/O, so focuses on CPU performance
  - Normalize relative to reference machine
  - Summarize as geometric mean of performance ratios
    - CINT2006 (integer) and CFP2006 (floating-point)

$$\sqrt[n]{\prod_{i=1}^{n} \text{Execution time ratio}_i}$$

CINT2006 for Intel Core i7 920

Description	Name	Instruction Count × 10^9	CPI	Clock cycle time (seconds × 10^{-9})	Executive Time (seconds)	Reference Time (seconds)	SPECratio
Interpreted string processing	perl	2,252	0.60	0.376	508	9,770	19.2
Block-sorting compression	bzip2	2,390	0.70	0.376	629	9,650	15.4
GNU C compiler	gcc	794	1.20	0.376	358	8,050	22.5
Combinatorial optimization	mcf	221	2.66	0.376	221	9,120	41.2
Go game (AI)	go	1,274	1.10	0.376	527	10,490	19.9
Search gene sequence	hmmer	2,616	0.60	0.376	590	9,330	15.8
Chess game (AI)	sjeng	1,948	0.80	0.376	586	12,100	20.7
Quantum computer simulation	libquantum	659	0.44	0.376	109	20,720	190.0
Video compression	h264avc	3,793	0.50	0.376	713	22,130	31.0
Discrete event simulation library	omnetpp	367	2.10	0.376	290	6,250	21.5
Games/path finding	astar	1,250	1.00	0.376	470	7,020	14.9
XML parsing	xalancbmk	1,045	0.70	0.376	275	6,900	25.1
Geometric Mean							25.7

Pitfall: Amdahl's Law

- Improving an aspect of a computer and expecting a proportional improvement in overall performance

$$T_{\text{improved}} = \frac{T_{\text{affected}}}{\text{improvement factor}} + T_{\text{unaffected}}$$

- Example: multiply accounts for 80s/100s
  - How much improvement in multiply performance to get 5× overall?

$$20 = \frac{80}{n} + 20$$

  - Can't be done!

- Corollary: make the common case fast

Example

- Use Amdahl's Law to compute the new execution time for an architecture that previously required 25 seconds to execute a program, where 15% of the time was spent executing load/store instructions, if the time required for a load/store operation is reduced by 40% (amount of improvement for load/stores = 1/.60 = 1.67).

Example

- Suppose you have a machine which executes a program consisting of 50% multiply instructions, 20% divide instructions, and the remaining 30% are other instructions. Management wants the machine to run 4 times faster. You can make the divide run at most 3 times faster and the multiply run at most 8 times faster. Can you meet management's goal by making only one improvement, and which one?

Multiprocessors

- Multicore microprocessors
  - More than one processor per chip
- Requires explicitly parallel programming
  - Compare with instruction level parallelism
    - Hardware executes multiple instructions at once
    - Hidden from the programmer
  - Hard to do
    - Programming for performance
    - Load balancing
    - Optimizing communication and synchronization

Example

- Assume the following instruction classes and corresponding CPIs and dynamic execution counts:

Type	CPI	Count
arithmetic	A	X
load/store	B	Y
branch	C	Z

When run on > 1 processors, the number of executed **arithmetic instructions** is divided by 0.7*p* (where *p* = number of processors) but the number of other instructions executed remains the same.

To what factor would the CPI of load/store instructions need to be reduced (sped up) in order for a **single processor** to match the performance of **four processors** each having the original CPI?

$$AX + \frac{BY}{f} + CZ = \frac{AX}{0.7 \cdot 4} + BY + CZ$$

Example

- Assume the following instruction classes and corresponding CPIs and dynamic execution counts:

Type	CPI	Count
arithmetic	A	X
load/store	B	Y
branch	C	Z

As compared to a single processor, under what condition is it possible to achieve an overall speedup of 6 by using multiple processors? Express this as an inequality.

$$\frac{1}{6} \geq \frac{\frac{A}{A+B+C}}{.7p} + \frac{B+C}{A+B+C}$$

Concluding Remarks

- Cost/performance is improving
  - Due to underlying technology development
- Hierarchical layers of abstraction
  - In both hardware and software
- Instruction set architecture
  - The hardware/software interface
- Execution time: the best performance measure
- Power is a limiting factor
  - Use parallelism to improve performance