U. Wisconsin CS/ECE 552
Introduction to Computer Architecture

Prof. Karu Sankaralingam

TAs: Taylor Johnston and Zhingyu Zhang

Welcome to 552
Prof. Karu is travelling,
so Taylor will intro the course today

Today

- Course overview and logistics
  - Syllabus
  - Course structure
- Introduction to Computer Architecture

Syllabus

- Language of the computer: ISA
- Arithmetic
- Processor Design
- Performance
- Memory
- IO
- Multiprocessors, Advanced processors, GPUs

Course Structure

- Lecture notes
  - Blackboard and slides
  - Some lectures will be “flipped” – video lectures available online
- Project
  - Describe a full processor, verify, and simulate it
  - Using a hardware description language called Verilog
  - *Optionally map to real hardware*
- Homework
- Two exams

Course Structure

- Lectures
  - Principles and Mechanisms
- Textbook:
  - D.A. Patterson and J.L. Hennessy, *Computer Architecture and Design: The Hardware/Software Interface*, 5th edition, Elsevier/Morgan Kauffman.
- Project:
  - Extensive use of design tools
  - Apply the principles and mechanisms to build a processor

Homework

- Homework 0: Introduce yourself
- Homework 1: Logic Design review
- Homework 2: Advanced Logic Design
- Homework 3: Intro to Verilog
- Homework 4: Advanced Verilog
- Homework 5: Intro to project
- Homework 6: Miscellaneous

- **One every two weeks**

Grading

- 20% Homework
- 30% Project
- 25% Midterm
- 25% Final

- Grades on Learn@UW

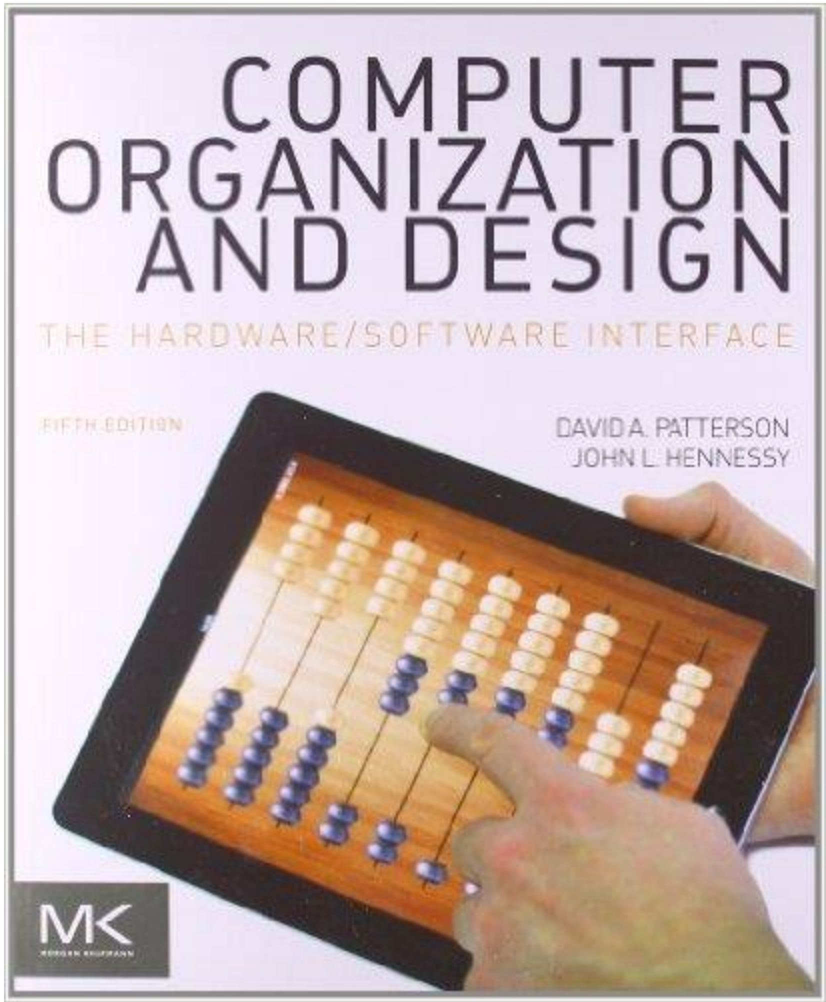
Online

- Web
  - http://www.cs.wisc.edu/~karu/courses/cs552
  - Course login and password
  - Course calendar, homework, lecture notes, and reference texts online
- Piazza for discussion:
  - TAs and I will look at it
- Email (we expect sparing usage)
  - Include 552 in subject of emails to me
  - Use Piazza if you think its of wide interest

Course Webpage

- Calendar
  - Home work date, exam dates, project deadline
- Homework
- Project
- Computing Tools

Required Text



Other

- Come to class on time
- Ask questions
- Submit homework on time
- Extensive office hours and feel free to drop in any time my office door is open

This is NOT an easy class

You will learn a lot – and will take a lot of time.

Miscellaneous Questions

- Mailing list mail?
- Piazza:
  - What is it?
  - Activated account?
- Who is not enrolled, but still wants to?
  - Write your name on the sheet

Topics you should be familiar with

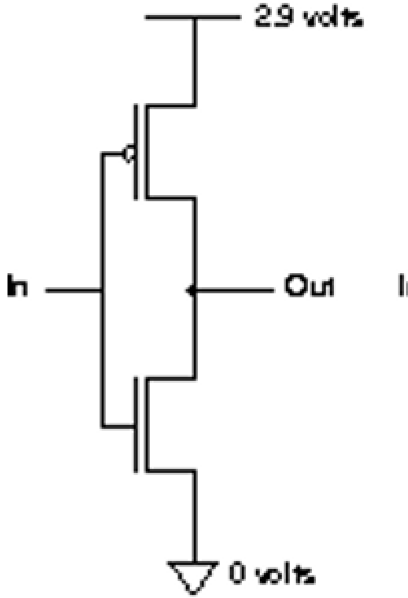
Boolean logic

$$X = \overline{A} \bullet \overline{B} \bullet C + \overline{A} \bullet B \bullet C + A \bullet \overline{B} \bullet C + A \bullet B \bullet \overline{C} + A \bullet B \bullet C$$

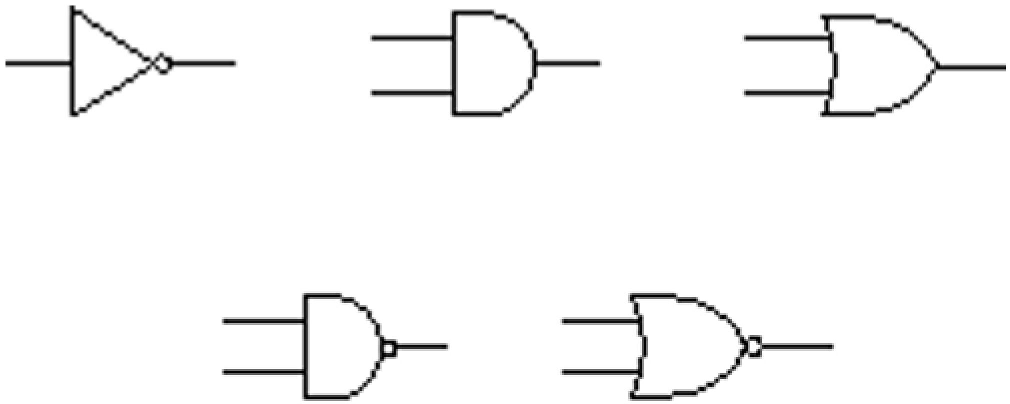
$$\begin{aligned}
&= \overline{A} \bullet C \bullet (\overline{B} + B) + A \bullet \overline{B} \bullet C + A \bullet B \bullet (\overline{C} + C) \\
&= \overline{A} \bullet C \bullet (1) + A \bullet \overline{B} \bullet C + A \bullet B \bullet (1) \\
&= \overline{A} \bullet C + A \bullet \overline{B} \bullet C + A \bullet B \\
&= \overline{A} \bullet C + A \bullet (\overline{B} \bullet C + B) \\
&= \overline{A} \bullet C + A \bullet (B + \overline{B}) \bullet (B + C) \\
&= \overline{A} \bullet C + A \bullet (1) \bullet (B + C) \\
&= \overline{A} \bullet C + A \bullet (B + C) \\
&= \overline{A} \bullet C + A \bullet B + A \bullet C \\
&= C \bullet (\overline{A} + A) + A \bullet B \\
&= C \bullet (1) + A \bullet B \\
&= A \bullet B + C
\end{aligned}$$

$$\begin{array}{cc}
A \cup A = A, & A \cap A = A, \\
A \cup A' = S, & A \cap A' = \phi, \\
A \cup B = B \cup A, & A \cap B = B \cap A, \\
A \cup S = S, & A \cap S = A, \\
A \cup \phi = A, & A \cap \phi = \phi, \\
A \cup (B \cup C) = (A \cup B) \cup C, & A \cap (B \cap C) = (A \cap B) \cap C, \\
A \cup (B \cap C) = (A \cup B) \cap (A \cup C), & A \cap (B \cup C) = (A \cap B) \cup (A \cap C), \\
(A \cup B)' = A' \cap B', & (A \cap B)' = A' \cup B',
\end{array}$$

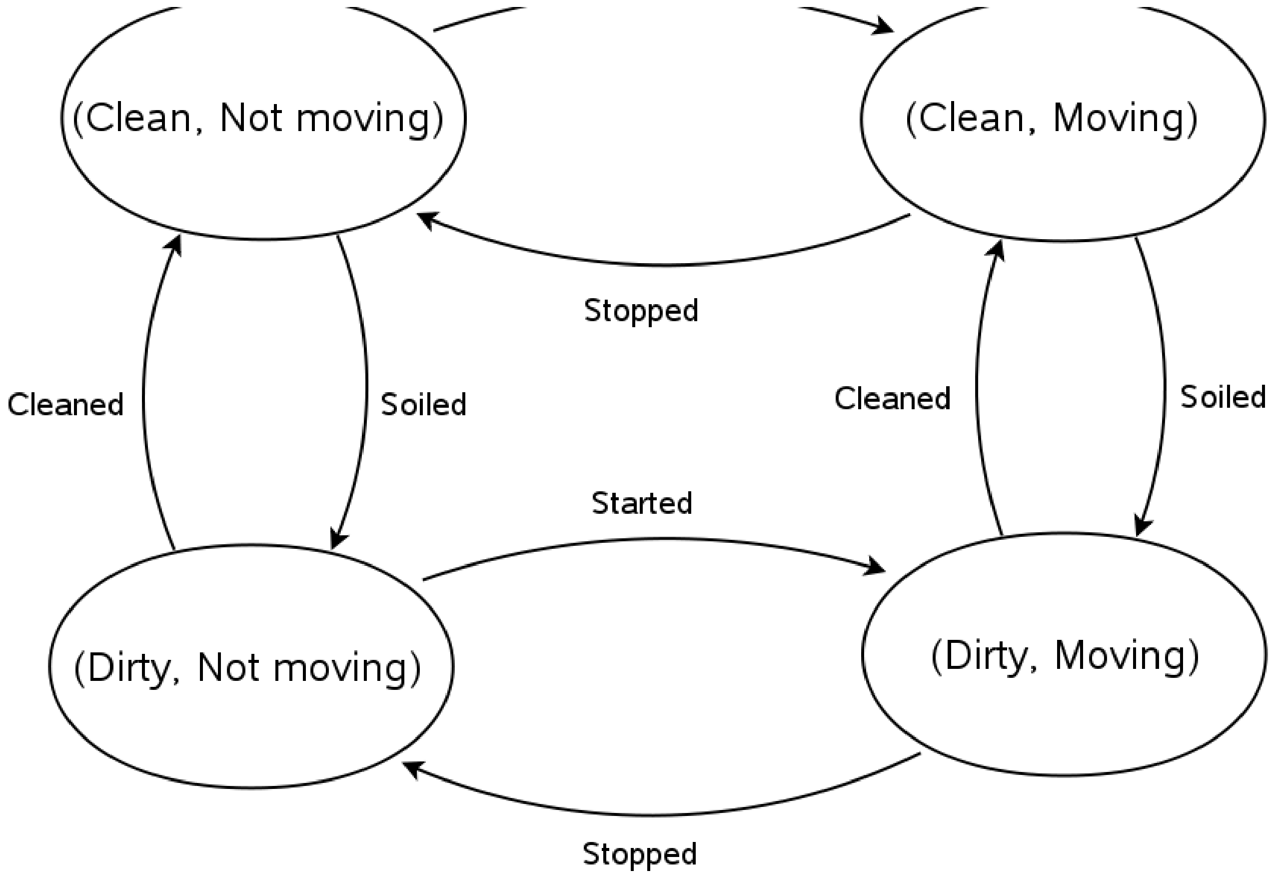
Transistors



Logic gates



State machines



Programming, c or java

```
#
main() {
    int c, first, last, middle, n, search, array[100];
    printf("Enter number of elements\n");
    scanf("%d",&n);
    printf("Enter %d integers\n", n);
    for ( c = 0 ; c < n ; c++ )
        scanf("%d",&array[c]);
    printf("Enter value to find\n");
    scanf("%d",&search);
    first = 0; last = n - 1; middle = (first+last)/2;
    ….
```

Assembly language

```
lw $t0, 4($gp)          # fetch N
mult $t0, $t0, $t0      # N*N
lw $t1, 4($gp)          #fetch N
ori $t2, $zero, 3       # 3
mult $t1, $t1, $t2      # 3*N
add $t2, $t0, $t1       # N*N + 3*N
sw $t2, 0($gp)          # i = ...
```

You DO NOT need to know Verilog

552 In Context

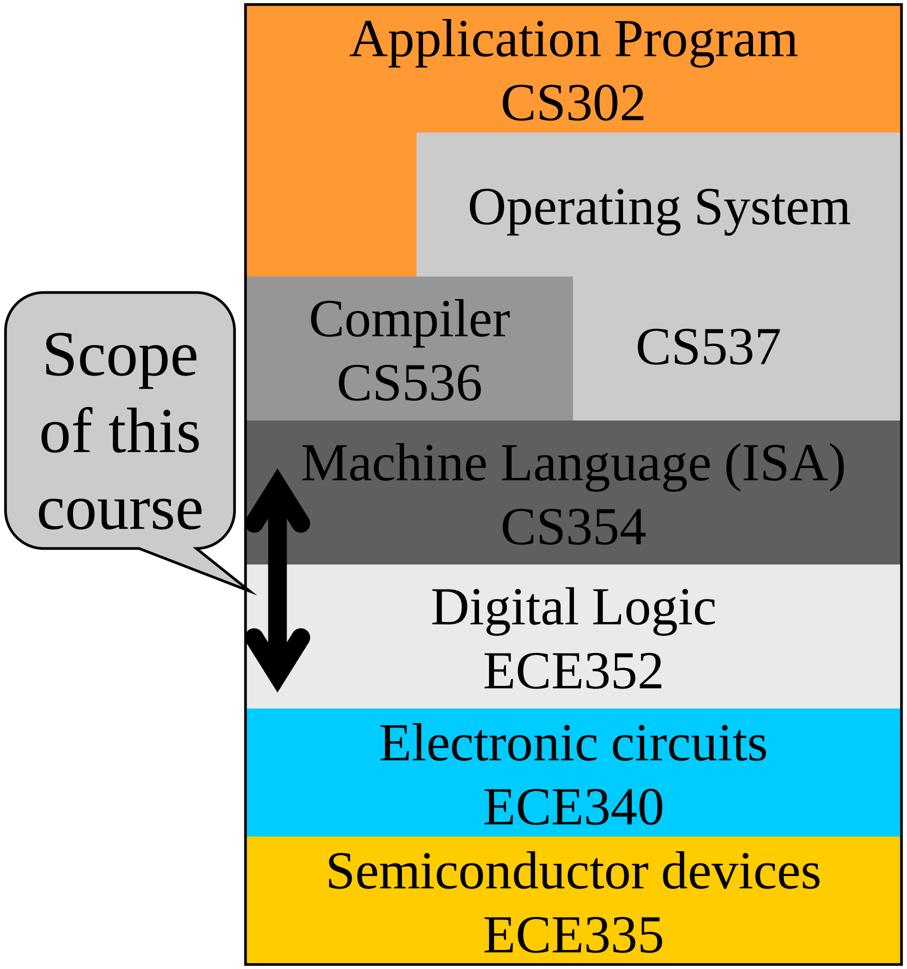
- Prerequisites
  - 252/352 – gates, logic, memory, organization
  - 252/354 – high-level language down to machine language interface or instruction set architecture (ISA)
- This course – 552 – puts it all together
  - Implement the logic that provides ISA interface
  - Must implement datapath and control
  - You will understand…no mystery
  - Manage tremendous complexity with abstraction

Why Take 552?

- To become a computer designer
  - Alumni of this class helped design your computer
- To learn what is *under the hood* of a computer
  - Innate curiosity
  - To write better code/applications
  - To write better system software (O/S, compiler, etc.)
- Because it is intellectually fascinating!
  - What is the most complex man-made device?

Abstraction and Complexity

- Abstraction helps us manage complexity
- Complex interfaces
  - Specify what to do
  - Hide details of how
- Goal: remove mystery



Computer Architecture

- Exercise in engineering tradeoff analysis
  - Find the fastest/cheapest/power-efficient/etc. solution
  - Optimization problem with 100s of variables
- All the variables are changing
  - At non-uniform rates
  - With inflection points
  - Only one guarantee: Today's right answer will be wrong tomorrow
- Two high-level effects:
  - Technology push
  - Application Pull

Technology Push

- What do these two intervals have in common?
  - 1947-1999 (53 years)
  - 2000-2001 (2 years)
- Answer: Equal progress in processor speed!
- The power of exponential growth!
- Driven by Moore's Law
  - Device per chips doubles every 18-24 months
- Computer architects work to turn the additional resources into speed/power savings/functionality!

Some History

Date	Event	Comments
1939	First digital computer	John Atanasoff (UW PhD ’30)
1947	1st transistor	Bell Labs
1958	1st IC	Jack Kilby (MSEE ’50) @TI Winner of 2000 Nobel prize
1971	1st microprocessor	Intel
1974	Intel 4004	2300 transistors
1978	Intel 8086	29K transistors
1989	Intel 80486	1.M transistors, pipelined
1995	Intel Pentium Pro	5.5M transistors
2005	Intel Montecito	1B transistors

Performance Growth

Unmatched by any other industry !

Doubling every 18 months (1982-1996): 800x
- Cars travel at 44,000 mph and get 16,000 mpg
- Air travel: LA to NY in 22 seconds (MACH 800)
- Wheat yield: 80,000 bushels per acre

- Doubling every 24 months (1971-1996): 9,000x
  - Cars travel at 600,000 mph, get 150,000 mpg
  - Air travel: LA to NY in 2 seconds (MACH 9,000)
  - Wheat yield: 900,000 bushels per acre

Technology Push

- Technology advances at varying rates
  - E.g. DRAM capacity increases at 60%/year
  - But DRAM speed only improves 10%/year
  - Creates gap with processor frequency!
- Inflection points
  - Crossover causes rapid change
  - E.g. enough devices for multicore processor (2001)
- Current issues causing an “inflection point”
  - Power consumption
  - Reliability
  - Variability

Application Pull

- Corollary to Moore's Law:
  Cost halves every two years

  In a decade you can buy a computer for less than its sales tax today. –Jim Gray

- Computers cost-effective for
  – National security – w
  – Enterprise
  – D ed design
  – eets, email, web
  – g – prescription drug labels

That was the old days….Now computers cost effective for even the most trivial of applications. What are the future applications? That is your job to figure out.

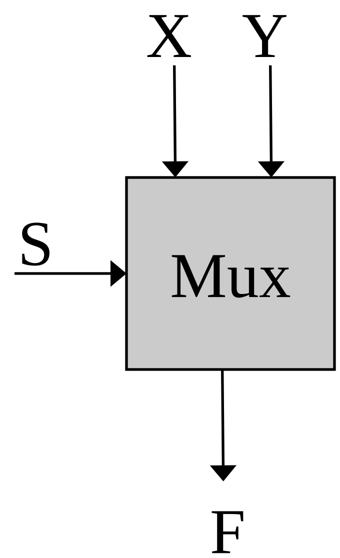
Abstraction

- Difference between interface and implementation
  - Interface: WHAT something does
  - Implementation: HOW it does so

- ***Career note****…Those who stay at the higher level with WHAT and don’t get too distracted by HOW have more successful long term engineering careers.*

Abstraction, E.g.

- 2:1 Mux (352)
- Interface

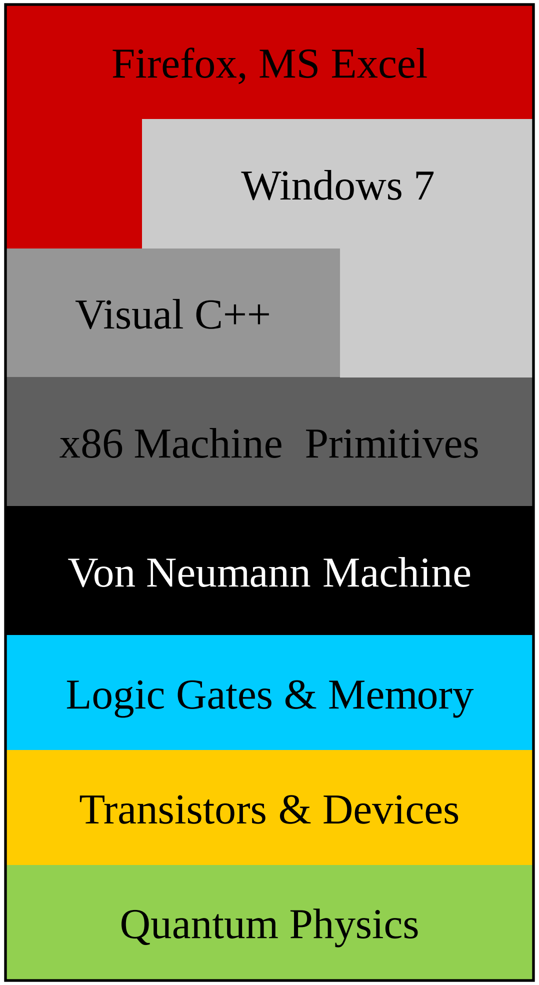


S	F
0	X
1	Y

- Implementations
  - Gates (fast or slow), pass transistors

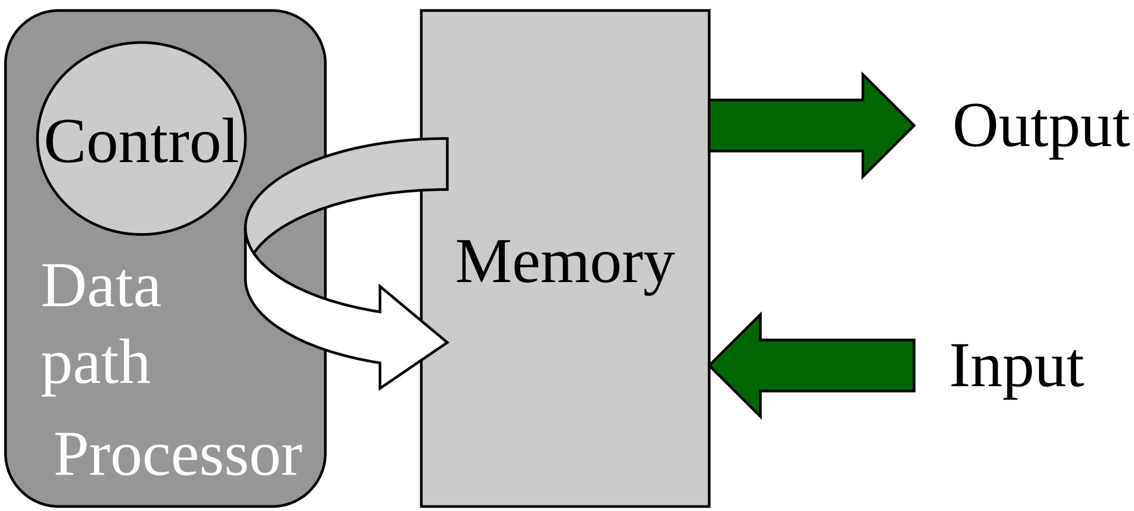
What's the Big Deal?

- Tower of abstraction
- Complex interfaces implemented by layers below
- Abstraction hides detail
- Hundreds of engineers build one product
- Complexity unmanageable otherwise



Basic Division of Hardware

- In space (vs. time)



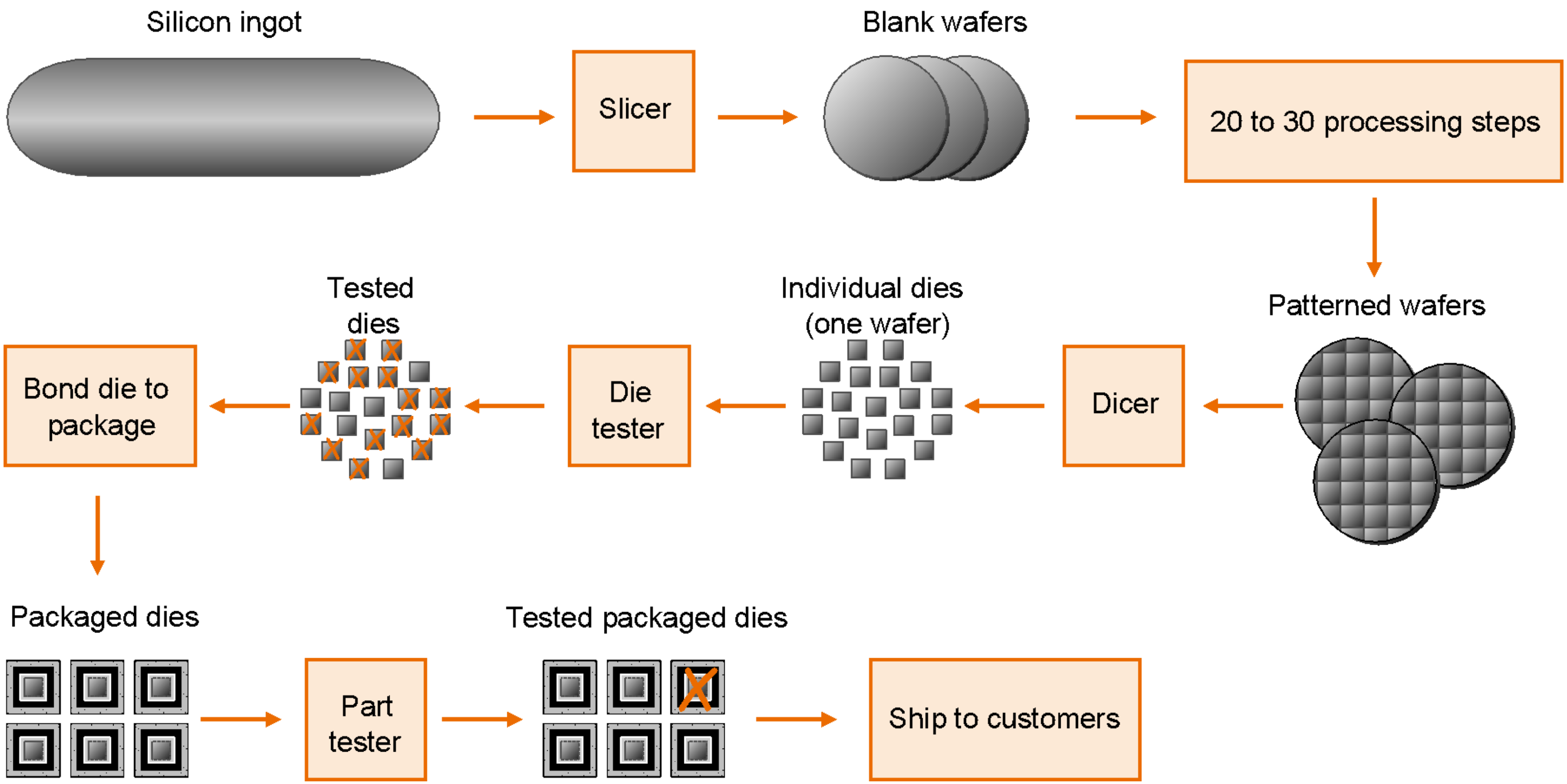
Basic Division of Hardware

- In time (vs. space)
  - Fetch instruction from memory **add r1, r2, r3**
  - Decode the instruction – what does this mean?
  - Read input operands read r2, r3
  - Perform operation add
  - Write results write to r1
  - Determine the next instruction pc := pc + 4

Building Computer Chips

- Complex multi-step process
  - Slice silicon ingots into wafers
  - Process wafers into patterned wafers
  - Dice patterned wafers into dies
  - Test dies, select good dies
  - Bond to package
  - Test parts
  - Ship to customers and make money

Building Computer Chips



Performance vs. Design Time

- Time to market is critically important
- E.g., a new design may take 3 years
  - It will be 3 times faster
  - But if technology improves 50%/year
  - In 3 years $1.5^3 = 3.38$
  - So the new design is worse!

    (unless it also employs new technology)

Bottom Line

- Designers must know BOTH software and hardware
- Both contribute to layers of abstraction
- IC costs and performance
- Compilers and Operating Systems

About This Course

- Course Textbook
  - D.A. Patterson and J.L. Hennessy, *Computer Architecture and Design: The Hardware/Software Interface*, 5th edition, Elsevier/Morgan Kauffman.

- Homework 0 due on Thursday
- Look at course calendar and project deadlines