ECE 550DK
Fundamentals of Computer Systems and Engineering

Fall 2023

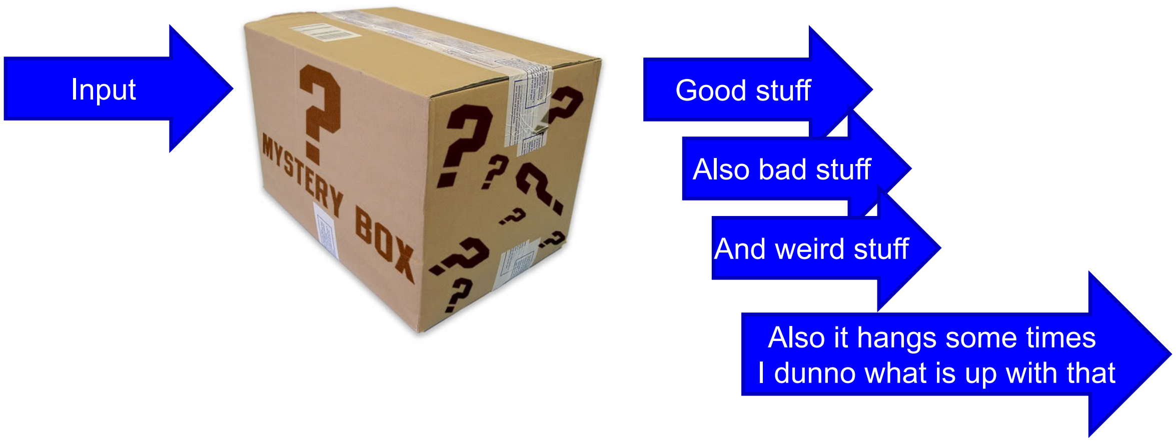
Introduction

Xin Li & Dawei Liu
Duke Kunshan University

Slides are derived from work by
Andrew Hilton, Tyler Bletsch and Rabih Younes (Duke)

Course objective:
Evolve your understanding of computers

Before



Input

Good stuff

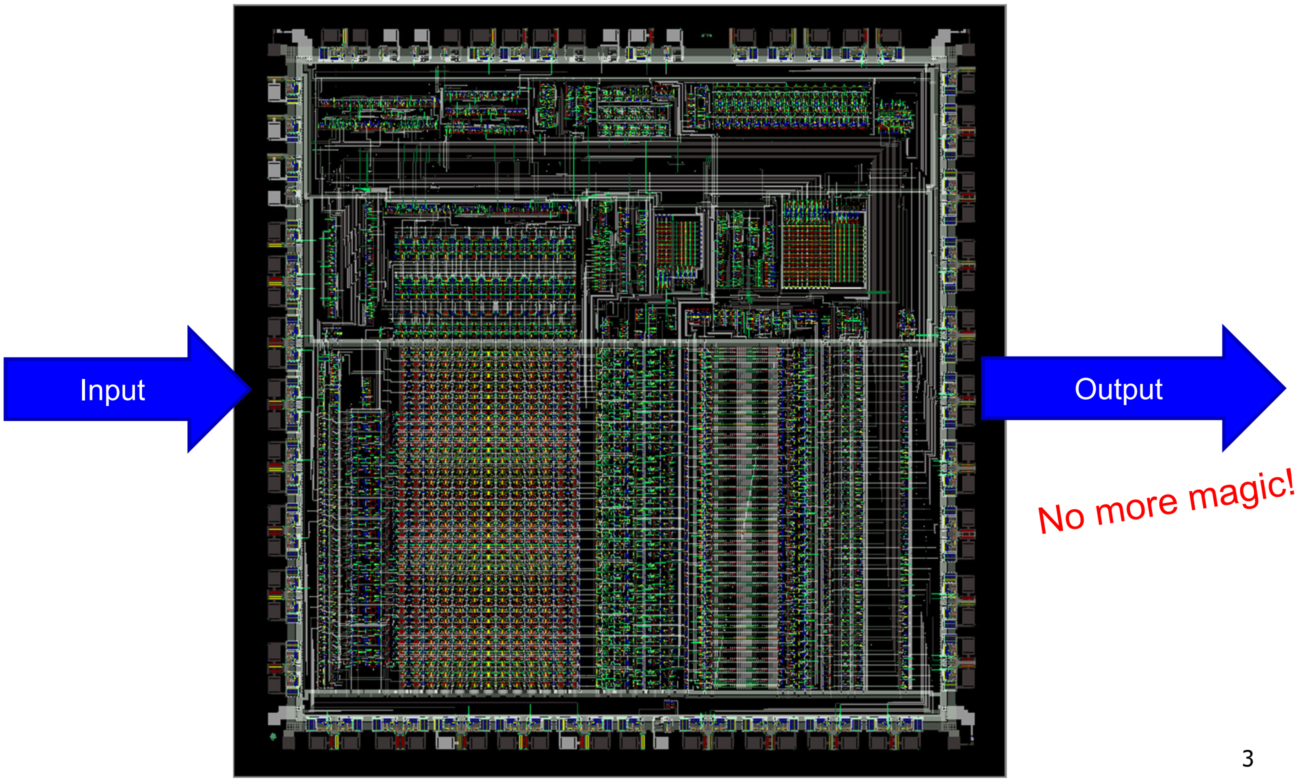
Also bad stuff

And weird stuff

Also it hangs some times
I dunno what is up with that

Course objective:
Evolve your understanding of computers

After



No more magic!

Overview

- For: MS/MEng students who want Comp Eng focus..
  - …but don't have Comp Eng undergrad
- Background for
  - **ECE 650: Systems Programming and Engineering**
    - ECE 558: Computer Networks/Distributed Systems
    - CS 510: Operating Systems
    - …
  - ECE 522: Advanced Computer Architecture
    - Co-req for Performance, Optimization, and Parallelism
    - Pre-req for 652
  - ECE 554: Fault-Tolerant and Testable Computer Systems
  - ECE 559: Advanced Digital System Design

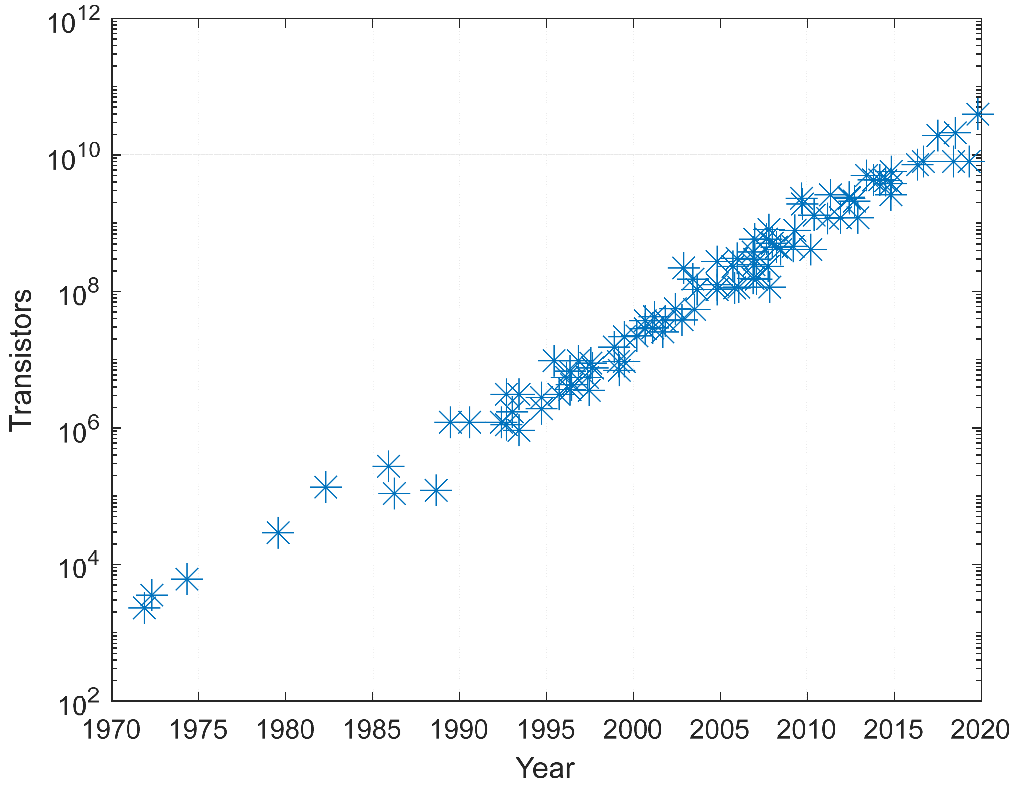
What we will learn: 10K feet

- Transistors -> Processor
  - Logic gates, combinational logic, sequential logic, FSMs
  - Adders, multipliers, shifters
  - Latches, Flip-flops, SRAMs, DRAMs, CAMs
  - Single-cycle datapaths, pipelining
  - Caches, memory hierarchy, virtual memory
  - Interrupts, exceptions, IO
- Hardware/software interface (ISA)
  - MIPS assembly

Transistors => Processors

- Processors are made of **transistors**
  - Electrically controlled switches (more on this later)
  - Anyone have any idea how many transistors are in a modern chip?

Microprocessor Trends



Transistors => Processors

- Processors are made of **transistors**
  - Electrically controlled switches (more on this later)
  - Anyone have any idea how many transistors are in a modern chip? **40 BILLION**
- How do you put together 40 Billion of anything
  - …and make sure the product works right?
  - …in every corner case
  - …and is really fast
  - …and do it within a reasonable budget/timeframe?
  - 1 transistor per second => 1268 years in total
- More fundamentally, how do you engineer any large system? **Abstraction**

Abstraction: The Key to Computer Engineering

- Abstraction: Divide interface from implementation
  - Interface: how its used
  - Implementation: how it does it
- Build larger components from smaller ones
  - Larger ones use interface of smaller ones to do tasks
  - Don't care about implementation
- Tasks can be split between engineers:
  - You make a piece that does xyz, and I'll use it to do my job
- Components can be re-used
  - Also good: making them generic, so they can be re-used more

Other key to engineering: tools

- Processors designed in Hardware Design Languages
  - Verilog
  - VHDL
  - Learn one: you can pick the other easily
- You don't lay out every transistor by hand…
  - Instead you write a description of the hardware in an HDL
  - …a lot like a programming language…
  - …then run it through synthesis tools
- We'll use Verilog and Quartus
  - With ModelSim to simulate

Levels of Abstraction

- Transistors: “electrical switch”
  - Can go lower (those with EE background have)
  - ..but no need for us
- Gates: a few transistors
  - Implement logical functions: And, Or, Nor, Xor
- Meaningful logic elements: a handful of gates
  - Combine into meaningful elements: muxes, 1-bit adders, flip-flops
  - May build larger items: N-bit adders from 1-bit adders
- Large elements (stages, units): combining logic elements
- Core
- Chip: now with multiple cores

A Different Kind of Abstraction

- Previous discussion: abstraction to build a processor

- Also: abstraction to **use** a processor
  - How/why?
  - Need software that can use the processor
  - Software should not rely on HOW processor is implemented
  - Abstraction between hardware implementation and interface
    - Interface = ISA = contract between hardware and software
    - Implementation: can vary from generation to generation
    - Consider x86
      - Can take a program written for an i386 (1985)
      - …and run it on a modern core in 2016

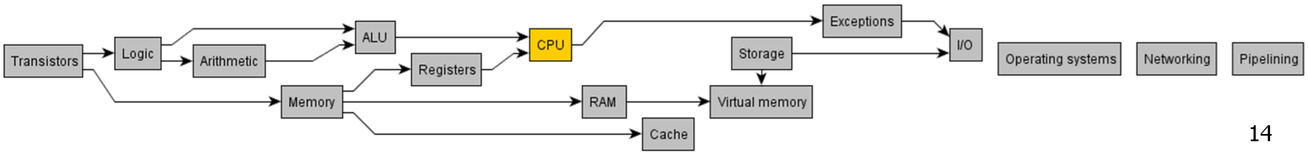
All computers are like fast food restaurants

- Fast Food Architecture: the interface
  - Menu
  - How/where to place orders
  - How finished orders are given to customers

- Fast Food Microarchitecture: the implementation
  - What ingredients are used
  - What appliances are available
  - How many employees you have and what they do

The ECE550 tech tree

1. We have **transistors**. They're electronic switches.
2. You can make **logic** by combining transistors.
3. You can make **arithmetic** from logic.
4. Transistors can also make basic **memory**.
5. Combine arithmetic and logic circuits to make an **Arithmetic/Logic Unit (ALU)**
6. Combine memory elements to make **registers** to store values
7. Combine the ALU, registers, and other stuff to make a full **CPU**!
8. Use other transistor configurations to make large-scale **RAM**.
9. RAM is too slow, so add **cache**, another form of memory to speed it up.
10. We have **storage** in the form of hard drives and SSDs.
11. Use storage and RAM together: this is **virtual memory**,
    which allows us to run more programs with more efficiency.
12. We halt the CPU with **exceptions** to handle things like **I/O** to storage.
13. The modern computer is governed by basic software called the **operating system**.
14. We have **networking** to connect multiple computers.
15. CPU performance is increased by various techniques, including **pipelining**.



Recitations

- Two sessions: Thursday
  - Only need to attend one of these two sessions
  - Please remember to attend the right session that you registered
  - Bring laptops
  - Mac people: get a Windows VM running
  - First recitation next week (no recitation this week)