Technology
Today

• Details about the course (assignments, tests, etc.)
• A brief history of architecture
• Technology building blocks
• Technology scaling
• Fundamentals of performance
Course Mechanics
Readings

- A Combination of the text and the research papers
- Papers
  - Fill out an on-line form for each paper.
  - The forms are on WebCT (linked off the course web page)
- The Text
  - The source of homework problems
Midterms and Final

• The midterms and final will focus more on analysis of the papers than the text.
  • Understand the main goal of the papers and their conclusions
  • Be prepared to think critically about the papers and synthesize ideas from them
  • If you give good, thoughtful answers to in the online form and pay attention/take good notes in class, you will probably do fine.
Prefetcher Competition

- If you don’t give an in-class presentation, you will compete in the prefetcher competition.
- Design and implement the best prefetcher you can
- There will be prizes!
- This involves reading several papers, and implementing your own prefetcher.
Assignments: Paper reading

- Read and *think about* each paper.
- Submit a summary.
- It is essential that you do this. Your grade depends substantially upon it.
- It is also essential that you learn to do this well.
  - Extracting content from papers is one of the most important skills in grad school.
Paper summaries

• Goal 1: Extract the good ideas from the paper.
  • This means discarding the junk.
  • Identifying the good parts.

• Goal 2: Understand how it fits into its context (i.e., the rest of architecture)
  • How is it similar/different/an extension of...?
What's the paper's goal?

- Does it solve a problem?
- Demonstrate an opportunity?
- Does it provide information?
What does it contribute?

• An idea?
• A mechanism?
• A description of an artifact?
• A methodology?
How do the authors substantiate their claims?

- Experiments?
- Real systems?
- Simulation?
- Prose arguments?
- Examples from “the real world”
How does the paper relate to others?

- Refute?
- Confirm?
- Extend?
- Synthesize?
- Re-examine?
  - In light of new tech./new app./new idea
What conclusions do they draw?

- Small conclusions
  - Did their idea work?
  - How well?
  - Do you believe them?

- Big conclusions
  - How do they think it should shape the future?
  - Do you believe them?
How well is the paper crafted?

• Does it tell a story?
• Is it interesting?
• Are the figures easy to understand?
• Do they properly highlight the important parts?
• Could you summarize the paper after looking at it for 5 minutes? (not for this class, you can’t ;-)
How would you improve the paper?

• Technically
  • Different approach (maybe you should write a paper?)
  • Methodology
  • Experiments

• Presentation
  • Organization
  • Additional background
  • Be concrete -- “make it more clear” is not useful.

• Be concrete -- “make it more clear” is not useful.
What questions does it raise?

- Issues with their approach?
- Directions for new work?
- Broader questions about architecture?
- What didn’t you understand?
Paper summaries

- Submitted via a WebCT form (see website)
- Due 10 minutes before class -- no exceptions.
  - You should never miss class for this
- You should bring a printed version of each paper to class!
In class presentation

• You will present one day’s worth of material in class
  • Become an expert on the topic.
  • Prepare 45-55 minutes of slides.
• Collect and answer questions for the next day.
Class presentation timeline

• 2 weeks ahead: Meet with me about the topic.
• 1 week ahead: Send me a draft of your slides
• Present your slides, collect questions.
• Prepare and present answers
• Send me slides with answers

• *You* are responsible for tracking these deadlines. Making them is part of your grade.
In Class Presentations

• There are 7 slots -- 2 people each.
• First come, first serve
  • Send email to myself and Hung Wei, to claim a date/topic.
• Once you’re signed up, you are committed.
• And you don’t have to do the prefetcher competition.
An Incomplete History of Computation
Charles Babbage 1791-1871
Lucasian Professor of Mathematics, Cambridge University, 1827-1839
First computer designer

Ada Lovelace 1815-1852
First computer programmer

Adapted from Arvind and Asanovic’s MIT course 6.823, Lecture 1
Charles Babbage

- **Difference Engine** 1823
  - The forerunner of modern digital computer!

- **Analytic Engine** 1833

**Application**
- Mathematical Tables – Astronomy
- Nautical Tables – Navy

**Background**
- Some efforts at mechanical calculators in the past.

**Technology**
- Mechanical - gears, Jacquard’s loom, simple calculators

Adapted from Arvind and Asanovic’s MIT course 6.823, Lecture 1
Difference Engine

1823
- Babbage’s paper is published

1834
- The paper is read by Scheutz & his son in Sweden

1842
- Babbage gives up the idea of building it; he is on to the Analytic Engine!

1855
- Scheutz displays his machine at the Paris World Fare
- Can compute any 6th degree polynomial
- Speed: 33 to 44 32-digit numbers per minute!

Adapted from Arvind and Asanovic’s MIT course 6.823, Lecture 1
Difference Engine

1823
- Babbage’s paper is published

1834
- The paper is read by Scheutz & his son in Sweden

1842
- Babbage gives up the idea of building it; he is on to the Analytic Engine!

1855
- Scheutz displays his machine at the Paris World Fare
- Can compute any 6th degree polynomial
- *Speed:* 33 to 44 32-digit numbers per minute!

*Now the machine is at the Smithsonian*

Adapted from Arvind and Asanovic’s MIT course 6.823, Lecture 1
Analytic Engine

The first conception of a general purpose computer

1. The store in which all variables to be operated upon, as well as all those quantities which have arisen from the results of the operations are placed.
2. The mill into which the quantities about to be operated upon are always brought.

An operation in the mill required feeding two punched cards and producing a new punched card for the store.

An operation to alter the sequence (i.e., a branch) was also provided!

Adapted from Arvind and Asanovic's MIT course 6.823, Lecture 1
Analytic Engine

1833: Babbage’s paper was published
   – conceived during a hiatus in the development of the difference engine

Inspiration: Jacquard Looms
   – looms were controlled by punched cards
     • The set of cards with fixed punched holes dictated the pattern of weave ⇒ program
     • The same set of cards could be used with different colored threads ⇒ numbers

1871: Babbage dies
   – The machine remains unrealized.

Adapted from Arvind and Asanovic’s MIT course 6.823, Lecture 1
Analytic Engine

1833: Babbage’s paper was published
- conceived during a hiatus in the development of the difference engine

Inspiration: Jacquard Looms
- looms were controlled by punched cards
  • The set of cards with fixed punched holes dictated the pattern of weave ⇒ program
  • The same set of cards could be used with different colored threads ⇒ numbers

1871: Babbage dies
- The machine remains unrealized.

It is not clear if the analytic engine could be built even today using only mechanical technology

Adapted from Arvind and Asanovic’s MIT course 6.823, Lecture 1
Linear Equation Solver
John Atanasoff, Iowa State University

1930’s:
– Atanasoff built the Linear Equation Solver.
– It had 300 tubes!

Application:
– Linear and Integral differential equations

Background:
– Vannevar Bush’s Differential Analyzer
  --- an analog computer

Technology:
– Tubes and Electromechanical relays
Linear Equation Solver
John Atanasoff, Iowa State University

1930’s:
– Atanasoff built the Linear Equation Solver.
– It had 300 tubes!

**Application:**
– Linear and Integral differential equations

**Background:**
– Vannevar Bush’s Differential Analyzer
  --- an analog computer

**Technology:**
– Tubes and Electromechanical relays

*Atanasoff decided that the correct mode of computation was by electronic digital means.*

Adapted from Arvind and Asanovic’s MIT course 6.823, Lecture 1
Harvard Mark I

- Built in 1944 in IBM Endicott laboratories
  - Howard Aiken – Professor of Physics at Harvard
  - Essentially mechanical but had some electromagnetically controlled relays and gears
  - Weighed 5 tons and had 750,000 components
  - A synchronizing clock that beat every 0.015 seconds

Adapted from Arvind and Asanovic’s MIT course 6.823, Lecture 1
Harvard Mark I

- Built in 1944 in IBM Endicott laboratories
  - Howard Aiken – Professor of Physics at Harvard
  - Essentially mechanical but had some electromagnetically controlled relays and gears
  - Weighed 5 tons and had 750,000 components
  - A synchronizing clock that beat every 0.015 seconds

Performance:
- 0.3 seconds for addition
- 6 seconds for multiplication
- 1 minute for a sine calculation
Harvard Mark I

• Built in 1944 in IBM Endicott laboratories
  – Howard Aiken – Professor of Physics at Harvard
  – Essentially mechanical but had some electromagnetically controlled relays and gears
  – Weighed 5 tons and had 750,000 components
  – A synchronizing clock that beat every 0.015 seconds

Performance:
  0.3 seconds for addition
  6 seconds for multiplication
  1 minute for a sine calculation

Broke down once a week!

Adapted from Arvind and Asanovic's MIT course 6.823, Lecture 1
Electronic Numerical Integrator and Computer (ENIAC)

- Inspired by Atanasoff and Berry, Eckert and Mauchly designed and built ENIAC (1943-45) at the University of Pennsylvania
- The first, completely electronic, operational, general-purpose analytical calculator!
  - 30 tons, 72 square meters, 200KW
- Performance
  - Read in 120 cards per minute
  - Addition took 200 $\mu$s, Division 6 ms
  - 1000 times faster than Mark I
- Not very reliable!

**WW-2 Effort**

**Application:** Ballistic calculations

angle = f (location, tail wind, cross wind, air density, temperature, weight of shell, propellant charge, ... )
Electronic Discrete Variable Automatic Computer (EDVAC)
Electronic Discrete Variable Automatic Computer (EDVAC)

• ENIAC’s programming system was external
  – Sequences of instructions were executed independently of the results of the calculation
  – Human intervention required to take instructions “out of order”
• Eckert, Mauchly, John von Neumann and others designed EDVAC (1944) to solve this problem
  – Solution was the stored program computer
    ⇒ “program can be manipulated as data”
• First Draft of a report on EDVAC was published in 1945, but just had von Neumann’s signature!
  – In 1973 the court of Minneapolis attributed the honor of inventing the computer to John Atanasoff

Adapted from Arvind and Asanovic’s MIT course 6.823, Lecture 1
Stored Program Computer

Program = A sequence of instructions

first stored program computer

Adapted from Arvind and Asanovic’s MIT course 6.823, Lecture 1
Stored Program Computer

Program = A sequence of instructions

How to control instruction sequencing?
Stored Program Computer

Program = A sequence of instructions

How to control instruction sequencing?

manual control

calculators

first stored program computer

Adapted from Arvind and Asanovic's MIT course 6.823, Lecture 1
Stored Program Computer

Program = A sequence of instructions

How to control instruction sequencing?

- manual control
- automatic control
  - external (paper tape)

calculators

Harvard Mark I, 1944

first stored program computer

Adapted from Arvind and Asanovic's MIT course 6.823, Lecture 1
Stored Program Computer

Program = A sequence of instructions

How to control instruction sequencing?

- manual control
- calculators
- automatic control
  - external (paper tape)
  - Harvard Mark I, 1944
  - Zuse’s Z1, WW2
  - internal
    - plug board
    - read-only memory
    - ENIAC 1946
    - read-write memory
    - ENIAC 1948
    - EDVAC 1947 (concept)

- The same storage can be used to store program and data

first stored program computer

Adapted from Arvind and Asanovic’s MIT course 6.823, Lecture 1
Store Program Computer

Program = A sequence of instructions

How to control instruction sequencing?

<table>
<thead>
<tr>
<th>manual control</th>
<th>calculators</th>
</tr>
</thead>
<tbody>
<tr>
<td>automatic control</td>
<td></td>
</tr>
<tr>
<td>external (paper tape)</td>
<td>Harvard Mark I, 1944</td>
</tr>
<tr>
<td>internal</td>
<td>Zuse’s Z1, WW2</td>
</tr>
<tr>
<td>plug board</td>
<td>ENIAC 1946</td>
</tr>
<tr>
<td>read-only memory</td>
<td>ENIAC 1948</td>
</tr>
<tr>
<td>read-write memory</td>
<td>EDVAC 1947 (concept)</td>
</tr>
</tbody>
</table>

- The same storage can be used to store program and data

EDSAC 1950 Maurice Wilkes

first stored program computer

Adapted from Arvind and Asanovic's MIT course 6.823, Lecture 1
And then there was IBM 701

IBM 701 -- 30 machines were sold in 1953-54

IBM 650 -- a cheaper, drum based machine, more than 120 were sold in 1954 and there were orders for 750 more! - eventually sold about 2000 of them

Adapted from Arvind and Asanovic's MIT course 6.823, Lecture 1
And then there was IBM 701

IBM 701 -- 30 machines were sold in 1953-54

IBM 650 -- a cheaper, drum based machine, more than 120 were sold in 1954 and there were orders for 750 more! - eventually sold about 2000 of them

Users stopped building their own machines.
And then there was IBM 701

IBM 701 -- 30 machines were sold in 1953-54

IBM 650 -- a cheaper, drum based machine, more than 120 were sold in 1954 and there were orders for 750 more! - eventually sold about 2000 of them

*Users stopped building their own machines.*

Why was IBM late getting into computer technology?
And then there was IBM 701

IBM 701 -- 30 machines were sold in 1953-54

IBM 650 -- a cheaper, drum based machine, more than 120 were sold in 1954 and there were orders for 750 more! - eventually sold about 2000 of them

Users stopped building their own machines.

Why was IBM late getting into computer technology?

*IBM was making too much money!* Even without computers, IBM revenues were doubling every 4 to 5 years in 40’s and 50’s.
Dominant Problem: Reliability

Mean time between failures (MTBF)

MIT’s Whirlwind with an MTBF of 20 min. was perhaps the most reliable machine!

Reasons for unreliability:

1. Vacuum Tubes
2. Storage medium
   acoustic delay lines
   mercury delay lines
   Williams tubes

Selections
- first cheap, reliable memory (~ 1 MHz)
- also called “core” (e.g., “core dump”)
- non volatile!
- destructive read cycle
- array of ferrite toroids (or “cores”)
- dominant memory technology until 70’s (→ ICs)

Adapted from Arvind and Asanovic’s MIT course 6.823, Lecture 1
Dominant Problem: Reliability

Mean time between failures (MTBF)

*MIT’s Whirlwind with an MTBF of 20 min. was perhaps the most reliable machine!*

Reasons for unreliability:

1. Vacuum Tubes
2. Storage medium
   - acoustic delay lines
   - mercury delay lines
   - Williams tubes

Selections

<table>
<thead>
<tr>
<th>Magnetic Core Memory</th>
<th>J. Forrester 1951</th>
</tr>
</thead>
<tbody>
<tr>
<td>- first cheap, reliable memory (~1 MHz)</td>
<td></td>
</tr>
<tr>
<td>- also called “core” (e.g., “core dump”)</td>
<td></td>
</tr>
<tr>
<td>- non volatile!</td>
<td></td>
</tr>
<tr>
<td>- destructive read cycle</td>
<td></td>
</tr>
<tr>
<td>- array of ferrite toroids (or “cores”)</td>
<td></td>
</tr>
<tr>
<td>- dominant memory technology until 70’s (→ ICs)</td>
<td></td>
</tr>
</tbody>
</table>

Adapted from Arvind and Asanovic’s MIT course 6.823, Lecture 1
Computers in mid 50’s
Computers in mid 50’s

- Hardware was expensive
Computers in mid 50’s

- Hardware was expensive
- Stores were small (1000 words)
Computers in mid 50’s

• Hardware was expensive
• Stores were small (1000 words)
  ⇒ No resident system-software!
Computers in mid 50’s

• Hardware was expensive
• Stores were small (1000 words)
  ⇒ No resident system-software!
• Memory access time was 10 to 50 times slower than the processor cycle
Computers in mid 50’s

• Hardware was expensive
• Stores were small (1000 words)  
  ⇒ No resident system-software!
• Memory access time was 10 to 50 times slower than the processor cycle  
  ⇒ Instruction execution time was totally dominated by the memory reference time.
Computers in mid 50’s

• Hardware was expensive
• Stores were small (1000 words)
  ⇒ No resident system-software!
• Memory access time was 10 to 50 times slower than the processor cycle
  ⇒ Instruction execution time was totally dominated by the memory reference time.
• The ability to design complex control circuits to execute an instruction was the central design concern as opposed to the speed of decoding or an ALU operation
Computers in mid 50’s

• Hardware was expensive
• Stores were small (1000 words)
  ⇒ No resident system-software!
• Memory access time was 10 to 50 times slower than the processor cycle
  ⇒ Instruction execution time was totally dominated by the memory reference time.
• The *ability to design complex control circuits* to execute an instruction was the central design concern as opposed to *the speed* of decoding or an ALU operation
• Programmer’s view of the machine was inseparable from the actual hardware implementation
Into the 60’s...:

Compatibility Problem at IBM
Compatibility Problem at IBM

By early 60’s, *IBM had 4 incompatible lines of computers!*

- 701 → 7094
- 650 → 7074
- 702 → 7080
- 1401 → 7010
Into the 60’s…:

Compatibility Problem at IBM

By early 60’s, *IBM had 4 incompatible lines of computers!*

701 → 7094  
650 → 7074  
702 → 7080  
1401 → 7010

Each system had its own

- Instruction set
- I/O system and Secondary Storage: magnetic tapes, drums and disks
- assemblers, compilers, libraries,…
- market niche
  business, scientific, real time, …

Adapted from Arvind and Asanovic’s MIT course 6.823, Lecture 3
Compatibility Problem at IBM

By early 60’s, *IBM had 4 incompatible lines of computers!*

701 → 7094
650 → 7074
702 → 7080
1401 → 7010

Each system had its own
- Instruction set
- I/O system and Secondary Storage: magnetic tapes, drums and disks
- assemblers, compilers, libraries, ...
- market niche
  business, scientific, real time, ...

⇒ *IBM 360*
IBM 360: Design Premises

Amdahl, Blaauw and Brooks, 1964


- Upward and downward, machine-language compatibility across a family of machines
- General purpose machine organization, general I/O interfaces, storage > 32K
- Easier to use (answers-per-month vs. bits-per-second)
- Machine must be capable of supervising itself without manual intervention → OS/360 (simple OS’s in IBM 700/7000)
- Built-in hardware fault checking and locating aids to reduce down time
- Simple to assemble systems with redundant I/O devices, memories etc. for fault tolerance

... the use of the “ISA” as a compatibility layer

$5 billion project (1964 dollars)

The Amdahl .. from Amdahl’s Law.
The Brooks .. from The Mythical Man-Month.
IBM 360: A General-Purpose Register (GPR) Machine

- Processor State
  - 16 General-Purpose 32-bit Registers
    - *may be used as index and base register*
    - *Register 0 has some special properties*
  - 4 Floating Point 64-bit Registers
  - A Program Status Word (PSW)
    - *PC, Condition codes, Control flags*
- A 32-bit machine with 24-bit addresses
  - No instruction contains a 24-bit address!
- Data Formats
  - 8-bit bytes, 16-bit half-words, 32-bit words, 64-bit double-words

Adapted from Arvind and Asanovic's MIT course 6.823, Lecture 3
IBM 360: Implementation

<table>
<thead>
<tr>
<th></th>
<th>Model 30</th>
<th>...</th>
<th>Model 70</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Storage</strong></td>
<td>8K - 64 KB</td>
<td></td>
<td>256K - 512 KB</td>
</tr>
<tr>
<td><strong>Datapath</strong></td>
<td>8-bit</td>
<td></td>
<td>64-bit</td>
</tr>
<tr>
<td><strong>Circuit Delay</strong></td>
<td>30 nsec/level</td>
<td></td>
<td>5 nsec/level</td>
</tr>
<tr>
<td><strong>Local Store</strong></td>
<td>Main Store</td>
<td></td>
<td>Transistor Registers</td>
</tr>
<tr>
<td><strong>Control Store</strong></td>
<td>Read only 1µsec (i.e. microcoded)</td>
<td></td>
<td>Conventional circuits</td>
</tr>
</tbody>
</table>

*IBM 360 instruction set architecture completely hid the underlying technological differences between various models.*

With minor modifications it survives till today

Adapted from Arvind and Asanovic's MIT course 6.823, Lecture 3
High performance competitor to IBM/S360

CDC 6600 Seymour Cray, 1964

a scientific supercomputer

Seymour’s 5P’s: Packaging, Plumbing (bits and heat flow), Parallelism, Programming, and understanding Programs

Adapted from Arvind and Asanovic's MIT course 6.823, Lecture 3
High performance competitor to IBM/S360

CDC 6600 Seymour Cray, 1964
a scientific supercomputer

- A fast pipelined machine with 60-bit words

Seymour’s 5P’s: Packaging, Plumbing (bits and heat flow), Parallelism, Programming, and understanding Programs

Adapted from Arvind and Asanovic's MIT course 6.823, Lecture 3
CDC 6600 Seymour Cray, 1964

a scientific supercomputer

- A fast pipelined machine with 60-bit words
- Ten functional units
  - Floating Point: adder, multiplier, divider
  - Integer: adder, multiplier
- Hardwired control (no microcoding)
- Dynamic scheduling of instructions using a scoreboard (see Appendix A)
- Ten Peripheral Processors for Input/Output
  - a fast time-shared 12-bit integer ALU

Seymour’s 5P’s: Packaging, Plumbing (bits and heat flow), Parallelism, Programming, and understanding Programs
CDC 6600 Seymour Cray, 1964

a scientific supercomputer

• A fast pipelined machine with 60-bit words
• Ten functional units
  - Floating Point: adder, multiplier, divider
  - Integer: adder, multiplier
  ...
• Hardwired control (no microcoding)
• Dynamic scheduling of instructions using a scoreboard (see Appendix A)
• Ten Peripheral Processors for Input/Output
  - a fast time-shared 12-bit integer ALU
• Very fast clock

Seymour’s 5P’s: Packaging, Plumbing (bits and heat flow), Parallelism, Programming, and understanding Programs

Adapted from Arvind and Asanovic’s MIT course 6.823, Lecture 3
High performance competitor to IBM/S360

CDC 6600  Seymour Cray, 1964

a scientific supercomputer

- A fast pipelined machine with 60-bit words
- Ten functional units
  - Floating Point: adder, multiplier, divider
  - Integer: adder, multiplier
- Hardwired control (no microcoding)
- Dynamic scheduling of instructions using a scoreboard (see Appendix A)
- Ten Peripheral Processors for Input/Output
  - a fast time-shared 12-bit integer ALU
- Very fast clock
- Novel freon-based technology for cooling

Seymour’s 5P’s: Packaging, Plumbing (bits and heat flow), Parallelism, Programming, and understanding Programs

Adapted from Arvind and Asanovic’s MIT course 6.823, Lecture 3
"Last week, Control Data ... announced the 6600 system. I understand that in the laboratory developing the system there are only 34 people including the janitor. Of these, 14 are engineers and 4 are programmers... Contrasting this modest effort with our vast development activities, I fail to understand why we have lost our industry leadership position by letting someone else offer the world's most powerful computer." -- T.J. Watson, IBM CEO

"It seems like Mr. Watson has answered his own question."

-- Seymour Cray
Learning more

• In 2006, UCSD, UW, and Berkeley ran a joint class on the history of computing
• It was really fantastic
  • It’s mostly lectures by the people who made the history
  • e.g. Steve Wozniak, Gordon Bell, Butler Lampson, Burton Smith,

• It’s all available online
  • http://www.cs.washington.edu/education/courses/csep590/06au/lectures/
  • Get the audio and listen to it.
Building Blocks: Logic, SRAMs, and DRAM
CMOS Logic Digression

- Complimentary Metal Oxide Semiconductor
  - Logic family (an abstraction!)

![Diagrams of PMOS and NMOS transistors with graphs showing current vs. gate voltage.](image_url)
CMOS Logic Digression

- Complimentary MOS

1.8V -- Logic One

0V -- Logic Zero
CMOS Logic Digression

- Complimentary MOS

1.8V -- Logic One

A = 1

Q = 1

1

0

A = 0

Q = 1

1

0

0V -- Logic Zero

A = 1

Q = 0

1

0

Pull-up network

Pull-down network
CMOS Logic Digression
CMOS Logic Digression
CMOS Logic Digression

- CMOS gates transfer current from Vdd to Vss via the output.

1.8V -- Logic One

- Current only flows when the output changes (almost)
  - Short circuit
  - Leakage

0V -- Logic Zero
CMOS Logic Digression

- CMOS gates transfer current from Vdd to Vss via the output.
  - Current only flows when the output changes (almost)
    - Short circuit
    - Leakage

1.8V -- Logic One

0V -- Logic Zero
CMOS Logic Digression

- CMOS gates transfer current from Vdd to Vss via the output.

1.8V -- Logic One

0V -- Logic Zero

- Current only flows when the output changes (almost)
  - Short circuit
  - Leakage

[Diagram showing CMOS gate with pull-up and pull-down networks]
CMOS Logic Digression

- CMOS gates transfer current from Vdd to Vss via the output.

  1.8V -- Logic One

  0V -- Logic Zero

- Current only flows when the output changes (almost)
  - Short circuit
  - Leakage

Pull-up network

Pull-down network
CMOS Logic Digression

- CMOS gates transfer current from Vdd to Vss via the output.

1.8V -- Logic One

0V -- Logic Zero

- Current only flows when the output changes (almost)
  - Short circuit
  - Leakage
CMOS Logic Digression

- CMOS gates transfer current from Vdd to Vss via the output.
  - Current only flows when the output changes (almost)
    - Short circuit
    - Leakage

1.8V -- Logic One
0V -- Logic Zero

![CMOS Logic Diagram](image)
CMOS Logic Digression

- CMOS gates transfer current from Vdd to Vss via the output.
  - Transitions take time (aka gate delay, RC delay, rise time)

1.8V -- Logic One

0V -- Logic Zero

\[ P = V \times I \]
\[ P = R \times I^2 \]
\[ I \sim \text{Load capacitance} \]
CMOS Logic Digression

- CMOS gates transfer current from Vdd to Vss via the output.

1.8V -- Logic One

0V -- Logic Zero

- Transitions take time (aka gate delay, RC delay, rise time)

\[ P = V \cdot I \]
\[ P = R \cdot I^2 + \text{Leakage} \]

I ~ Load capacitance
CMOS Logic Digression

- Leakage is a growing problem, but it’s a settable parameter.
  - We can limit it, but it impacts speed.
  - 30% of power in leakage seems like a good balance point in practice.

Source: Intel
• Leakage is a growing problem, but it’s a settable parameter.
  – We can limit it, but it impacts speed.
  – 30% of power in leakage seems like a good balance point in practice.

Source: Intel
CMOS Logic Digression

• Leakage is a growing problem, but it’s a settable parameter.
  – We can limit it, but it impacts speed.
  – 30% of power in leakage seems like a good balance point in practice.

Circuits dissipate power even when not switching

Source: Intel
Silicon Memories

• Why store things in silicon?
  • It’s fast!!!
  • Compatible with logic devices (mostly)
• The main goal is to be cheap
  • Dense -- The smaller the bits, the less area you need, and
    the more bits you can fit on a chip/wafer/through your fab.
  • Bit sizes are measured in $F^2$ -- the smallest feature you can
    create.
  • $F^2$ is a function of the memory technology, not the
    manufacturing technology.
Questions

• What physical quantity should represent the bit?
  • Voltage/charge -- SRAMs, DRAMs, Flash memories
  • Magnetic orientation -- MRAMs
  • Crystal structure -- phase change memories
  • The orientation of organic molecules -- various exotic technologies
  • All that’s required is that we can sense it and turn it into a logic one or zero.

• How do we achieve maximum density?
• How do we make them fast?
Anatomy of a Memory

- **Dense**: Build a big array
  - bigger the better
  - less other stuff
  - Bigger -> slower
- **Row decoder**
  - Select the row by raising a “word line”
- **Column decoder**
  - Select a slice of the row
- **Decoders are pretty big.**
The Storage Array

• Density is king.
  • Highly engineered, carefully tuned, automatically generated.
  • The smaller the devices, the better.
• Making them big makes them slow.
  • Bit/word lines are long (millimeters)
  • They have large capacitance, so their RC delay is long
  • For the row decoder, use large transistors to drive them hard.
• For the bit cells...
  • There are lots of these, so they need to be as small as possible (but not smaller)
Sense Amps

- Sense amplifiers take a difference between two signals and amplify it.
- Two scenarios:
  - Inputs are initially equal ("precharged") -- they each move in opposite directions.
  - One input is a reference -- so only one signal moves.
- Frequently used in memories:
  - Sense amps can detect small analog signals from the storage cell, and convert it into a logic one or logic zero.
Static Random Access Memory (SRAM)

- **Storage**
  - Voltage on a pair of cross-coupled inverters
  - Durable in presence of power
- **To read**
  - Pre-charge two bit lines to Vcc/2
  - Turn on the “word line”
  - Read the output of the sense-amp
Static Random Access Memory (SRAM)

- **Storage**
  - Voltage on a pair of cross-coupled inverters
  - Durable in presence of power
- **To read**
  - Pre-charge two bit lines to Vcc/2
  - Turn on the “word line”
  - Read the output of the sense-amp
Static Random Access Memory (SRAM)

- **Storage**
  - Voltage on a pair of cross-coupled inverters
  - Durable in presence of power

- **To read**
  - Pre-charge two bit lines to Vcc/2
  - Turn on the “word line”
  - Read the output of the sense-amp
Static Random Access Memory (SRAM)

- **Storage**
  - Voltage on a pair of cross-coupled inverters
  - Durable in presence of power

- **To read**
  - Pre-charge two bit lines to Vcc/2
  - Turn on the “word line”
  - Read the output of the sense-amp
Static Random Access Memory (SRAM)

- **Storage**
  - Voltage on a pair of cross-coupled inverters
  - Durable in presence of power
- **To read**
  - Pre-charge two bit lines to Vcc/2
  - Turn on the “word line”
  - Read the output of the sense-amp
SRAM Writes

- To write
  - Turn off the sense-amp
  - Turn on the wordline
  - Drive the bitlines to the correct state
  - Turn off the wordline
SRAM Writes

• To write
  • Turn off the sense-amp
  • Turn on the wordline
  • Drive the bitlines to the correct state
  • Turn off the wordline
SRAM Writes

• To write
  • Turn off the sense-amp
  • Turn on the wordline
  • Drive the bitlines to the correct state
  • Turn off the wordline
SRAM Writes

- To write
  - Turn off the sense-amp
  - Turn on the wordline
  - Drive the bitlines to the correct state
  - Turn off the wordline
SRAM Writes

- To write
  - Turn off the sense-amp
  - Turn on the wordline
  - Drive the bitlines to the correct state
  - Turn off the wordline
Building SRAM

- This is “6T SRAM”
- 6 “basic devices” is pretty big
- SRAMs are not dense
SRAM Density

- At 65nm: $0.52\text{um}^2$
- $123-140\text{ F}^2$
- $1\text{ F}^2$ is one “square feature”
- [ITRS 2008]

65nm TSMC 6T SRAM
SRAM Ports

- Add word and bit lines
- Read/write multiple things at once
- Density decreases quadratically
- Bandwidth increase linearly
SRAM Performance

- Read and write times
  - 10s-100s of ps

- Bandwidth
  - Registers -- 324GB/s
  - L1 cache -- 128GB/s
  - Samsung K7D323674C -- 3.6GB/s

- Durability
  - Infinite (not quite actually, but very close)
SRAM’s future

• SRAM is a mature technology. No new, big breakthroughs or advances are expected beyond CMOS scaling.
Dynamic Random Access Memory (DRAM)

- **Storage**
  - Charge on a capacitor
  - Decays over time (us-scale)
  - This is the “dynamic” part.
  - About $6F^2$: 20x better than SRAM

- **Reading**
  - Precharge
  - Assert word line
  - Sense output
  - Refresh data
Dynamic Random Access Memory (DRAM)

- **Storage**
  - Charge on a capacitor
  - Decays over time (us-scale)
  - This is the “dynamic” part.
  - About $6F^2$: 20x better than SRAM

- **Reading**
  - Precharge
  - Assert word line
  - Sense output
  - Refresh data

![DRAM Circuit Diagram]
Dynamic Random Access Memory (DRAM)

- **Storage**
  - Charge on a capacitor
  - Decays over time (us-scale)
  - This is the “dynamic” part.
  - About $6F^2$: 20x better than SRAM

- **Reading**
  - Precharge
  - Assert word line
  - Sense output
  - Refresh data
Dynamic Random Access Memory (DRAM)

- **Storage**
  - Charge on a capacitor
  - Decays over time (us-scale)
  - This is the “dynamic” part.
  - About $6F^2$: 20x better than SRAM

- **Reading**
  - Precharge
  - Assert word line
  - Sense output
  - Refresh data
Dynamic Random Access Memory (DRAM)

- **Storage**
  - Charge on a capacitor
  - Decays over time (us-scale)
  - This is the “dynamic” part.
  - About $6F^2$: 20x better than SRAM

- **Reading**
  - Precharge
  - Assert word line
  - Sense output
  - Refresh data

---

![Diagram of DRAM reading process](image)
Dynamic Random Access Memory (DRAM)

- **Storage**
  - Charge on a capacitor
  - Decays over time (us-scale)
  - This is the “dynamic” part.
  - About 6F²: 20x better than SRAM

- **Reading**
  - Precharge
  - Assert word line
  - Sense output
  - Refresh data
Dynamic Random Access Memory (DRAM)

- **Storage**
  - Charge on a capacitor
  - Decays over time (us-scale)
  - This is the “dynamic” part.
  - About $6F^2$: 20x better than SRAM

- **Reading**
  - Precharge
  - Assert word line
  - Sense output
  - Refresh data
DRAM: Write and Refresh

- **Writing**
  - Turn on the wordline
  - Override the sense amp.

- **Refresh**
  - Every few micro-seconds, read and re-write every bit.
  - Consumes power
  - Takes time
DRAM Lithography
DRAM Devices

- There are many banks per die (16 at left)
  - Multiple can be active at once to hide latencies
  - Parallelism!!!

- Example
  - open bank 1, row 4
  - open bank 2, row 7
  - open bank 3, row 10
  - read bank 1, column 8
  - read bank 2, column 32
  - ...

Micron 78nm 1Gb DDR3
DRAM: Micron
MT47H512M4
DRAM: Micron
MT47H512M4
DRAM Variants

- The basic DRAM technology has been wrapped in several different interfaces.
- SDRAM (synchronous)
- DDR SDRAM (double data-rate)
  - Data clocked on rising and falling edge of the clock.
- DDR2
  - Example on previous slides
- DDR3
- RDRAM
  - Rambus RAM
- GDDR2-5 -- For graphics cards.
- FB-DIMMS
DDR2 SDRAM

- Fewer, larger banks.
- Pin count per package (DIMM): 14 address, 16 data
- DIMM data path is 64 bits
- Data rate: up to 800 MHz DDR (1600 MHz effective)
- Bandwidth per DIMM GTNE: 12.8 GB/s
  - guaranteed not to exceed
- Multiple DIMMs can attach to a bus
  - Reduces bandwidth/GB (a good idea?)
Power

- DRAM is a major power sink.
- Idle power: 2-4W/DIMM
- Active power: 5-8W/DIMM

Economou, et. al 2006
DRAM Scaling

- Long term need for performance has driven DRAM hard
  - complex interface.
  - High performance
  - High power.
- DRAM used to be the main driver for process scaling, now it’s flash.
- Scaling is expected to match CMOS tech scaling
- $F^2$ cell size will probably not decrease
Technology Scaling
Moore’s Law: 2X transistors / “year”

“Cramming More Components onto Integrated Circuits”
– Gordon Moore, Electronics, 1965

# on transistors / cost-effective integrated circuit double every N months (12 ≤ N ≤ 24)

Adapted from Patterson, CSE 252 Sp06 Lecture 2 © 2006 UC Berkeley.
Moore’s Law: 2X transistors / “year”

- Gordon Moore, Electronics, 1965

# on transistors / cost-effective integrated circuit double every N months (12 ≤ N ≤ 24)

N seems to be rising - now 30?

“Cramming More Components onto Integrated Circuits”

Adapted from Patterson, CSE 252 Sp06 Lecture 2 © 2006 UC Berkeley.
The essence of Moore’s Law:

scale each dimension by \( \sim 1/\sqrt{2} = \sim 0.71 \)

each scaling halves the area of a fixed design

“Process” or “Lithography” or “Litho” Generation
smallest wire pitch = \( \sim 2-3 \times \) litho

180 nm \( \rightarrow \) 130 nm \( \rightarrow \) 90 nm
The same design is often shrunk through multiple process generations before coming up with a new micro-architecture, which is adjusted for technology changes.
Because chip costs scale with around the square of die size, there is a “target die size” …
Since technology change is such a big influence in architecture, and because it takes 3-6 years to create a totally new design, we try to predict & exploit it (with varying degrees of success.)
**Transistor Frequency Scaling**

*Transistor* (not processor) frequency scales approximately linearly with feature size (e.g., 1.4x/generation), est. 17%/year.

So where did the remainder of the 39% per year and 58% per year come from?

**Diagram:**
- CISC-on-RISC
  - Speculation to reduce critical paths
  - More pipelining
  - Faster arithmetic structures
  - Aggressive logic families (e.g., dual rail domino)

**Categories:**
- Language
- Compiler
- "ISA"
- Micro Architecture
- RTL
- Circuits
- Devices
- Materials Science

17%
Transistor (not processor) frequency scales approximately linearly with feature size. (e.g., 1.4x / generation), est. 17%/year.

So where did the remainder of the 39% per year and 58% per year come from?

- CISC-on-RISC
- speculation to reduce critical paths
- more pipelining
- faster arithmetic structures
- aggressive logic families (e.g., dual rail domino)
- 17%
- the power wall - has reduced ability of these things to further improve frequency

Language
Compiler
“ISA”
Micro Architecture
RTL
Circuits
Devices
Materials Science
Computer Performance
Computer Performance

Relative Performance vs. Year

- specINT95
- specINT2000
- specINT2006

47% per year
39% per year
Computer Performance

Relative Performance

Year

specINT95
specINT2000
specINT2006

47% per year
39% per year
25% per year
What determines the cycle time today?

• Old days: 10-16 levels of gates
• Today: gate delays, clock overhead, wire delays, + POWER

Adapted from Patterson, CSE 252 Sp06 Lecture 1 © 2006 UC Berkeley.
Processor Frequency and Power

P = V^2 FCa
V = Processor voltage
F = frequency
C = Output load
a = Activity factor

- Lower frequency
  - Less overhead
  - Linear power savings
  - Potentially lower voltage -> quadratic power savings
  - More layers of logic per clock
  - Performance impact?