

CS 110
Computer Architecture
Lecture 30:
Course Summary
Video 1: Admin

Instructors:

Sören Schwertfeger & Chundong Wang

<https://robotics.shanghaitech.edu.cn/courses/ca/20s/>

School of Information Science and Technology SIST

ShanghaiTech University

Slides based on UC Berkley's CS61C

Quiz on Security

Piazza: “Video Lecture 29 Security”

- Which one of the following statements is **NOT** true?
 - A. In a processor with inclusive cache, Flush+Reload will flush a cache line from all levels of CPU cache.
 - B. Monitoring network traffic may provide a side channel to attack cloud computing applications.
 - C. Smartphones are not affected by Rowhammer because the DRAM chips in them are battery-backed.
 - D. Using Meltdown to dump the entire kernel memory does not require any permission or privilege.

Quiz on Security

Piazza: “Video Lecture 29 Security”

- Which one of the following statements is **NOT** true?
 - A. In a processor with inclusive cache, Flush+Reload will flush a cache line from all levels of CPU cache.
 - B. Monitoring network traffic may provide a side channel to attack cloud computing applications.
 - C. Smartphones are not affected by Rowhammer because the DRAM chips in them are battery-backed.
 - D. Using Meltdown to dump the entire kernel memory does not require any permission or privilege.

Final

- Date: Tuesday, June 23rd, 2020
- Time: 8:00 - 10:00 (normal lecture slot++)
 - Be there latest 7:45 – **we start 8:00 sharp!**
- Venue: 3 rooms – check on egate which room you are!
 - 教学中心201
 - 教学中心202
 - 教学中心203
- Closed book:
 - You can bring **three** A4 pages with notes (both sides; in **English**): Write your Chinese and **Pinyin** name on the top!
Handwritten by you!
 - You will be provided with the RISC-V "green sheet"
 - No other material allowed!

Final

- Wear your Corona mask!
- Switch cell phones **off!**
(not silent mode – off!)
 - Put them in your bags.
- Bags under the table. Nothing except paper, pen, 1 drink, 1 snack, your student ID card on the table!
- No other electronic devices are allowed!
 - No ear plugs, music, smartwatch...
- Anybody touching any electronic device will **FAIL** the course!
- Anybody found cheating (copy your neighbors answers, additional material, ...) will **FAIL** the course!
- Content: Everything!

Next Lecture

- Next Online Lecture:
 - Q&A!
 - Prepare your questions
 - We will try to answer live – or on piazza

Admin



Admin



Admin



Admin

COMPUTER ORGANIZATION AND DESIGN

THE HARDWARE/SOFTWARE INTERFACE

 RISC-V EDITION



MK
MORGAN KAUFMANN

DAVID A. PATTERSON
JOHN L. HENNESSY

FUNCTIONS OF SEVERAL VARIABLES	$z = f(x, y)$ or $f(x, y, z)$	DOMAIN: Allowed (x, y) , (x, y, z) RANGE: z or f
LEVEL-CURVES (2-D) $z = f(x, y) = k = \text{CONST.}$ CONTOUR MAPS (2-D) $\Delta z = f(x, y) = k = \text{CONST.}$ SURFACE LAYERS (3-D)	FUNCTIONS OF N VARIABLES $z = f(x_1, x_2, \dots, x_n)$ or $f(x_1, \dots, x_n)$ DOMAIN: Allowed (x_1, \dots, x_n) RANGE: z or f IF FOR EVERY $W \in \mathbb{R}$, THERE IS A CORRESPONDING (x_1, \dots, x_n) SUCH THAT $f(x_1, \dots, x_n) = W$, THEN f IS ONTO. IF FOR EVERY $W \in \mathbb{R}$, THERE IS A CORRESPONDING (x_1, \dots, x_n) SUCH THAT $f(x_1, \dots, x_n) = W$, THEN f IS ONTO.	P-D DEFINITIONS OF CONTINUITY LET f BE A FUNCTION OF N VARIABLES DEFINED ON A SET D OF POINTS IN \mathbb{R}^n . EXCEPT POSSIBLY AT (a, b) , THEN f IS CONTINUOUS AT (a, b) IF FOR EVERY $\epsilon > 0$, THERE IS A $\delta > 0$ SUCH THAT IF $\ (x, y) - (a, b)\ < \delta$, THEN $ f(x, y) - f(a, b) < \epsilon$.
PARTIAL DERIVATIVES $z = f(x, y)$ $f_x(x, y) = \frac{\partial f}{\partial x}$ $f_y(x, y) = \frac{\partial f}{\partial y}$	DERIVATIVES DERIVATIVE OF RESPECT TO ONE VARIABLE, WHILE HOLDING THE OTHER VARIABLES CONSTANT. SAME RULES FOR FUNCTIONS OF MORE THAN TWO VARIABLES.	IF THE LIMIT AS A POINT (a, b) ALONG TWO DIFFERENT PATHS IS NOT THE SAME, THE LIMIT DOES NOT EXIST. $f(x, y)$ IS CONTINUOUS AT (a, b) IF THE LIMIT OF $f(x, y)$ AS $(x, y) \rightarrow (a, b)$ EXISTS.
SECOND PARTIAL DERIVATIVES $f_{xx} = \frac{\partial^2 f}{\partial x^2}$ $f_{yy} = \frac{\partial^2 f}{\partial y^2}$ $f_{xy} = \frac{\partial^2 f}{\partial x \partial y}$ $f_{yx} = \frac{\partial^2 f}{\partial y \partial x}$	CLAIRAUT'S THEOREM IF f_{xy} AND f_{yx} ARE BOTH CONTINUOUS AT (a, b) , THEN $f_{xy}(a, b) = f_{yx}(a, b)$.	COMPOSITE FUNCTIONS OF CONTINUOUS FUNCTIONS ARE CONTINUOUS, AS ARE SUMS AND PRODUCTS EQUATIONS OF TANGENT PLANES TO SURFACES $z = f(x, y)$ AT POINT (x_0, y_0, z_0) $z - z_0 = f_x(x_0, y_0)(x - x_0) + f_y(x_0, y_0)(y - y_0)$
THE CHAIN RULE $\frac{dz}{dt} = \frac{\partial z}{\partial x} \frac{dx}{dt} + \frac{\partial z}{\partial y} \frac{dy}{dt}$	PARTIAL DIFF. EQS LAPLACE'S EQUATION: $\nabla^2 f = 0$ HELMHOLTZ EQUATION: $\nabla^2 f = -g$	TOTAL DIFFERENTIAL $dz = f_x dx + f_y dy + f_z dz$ INCREMENTS $\Delta x, \Delta y, \Delta z$ DIFFERENTIALS dx, dy, dz IF f IS DIFFERENTIABLE AT (a, b) , THEN $\Delta f \approx df$ AS $(\Delta x, \Delta y) \rightarrow (0, 0)$.
CASES $\frac{dz}{dt} = \frac{\partial z}{\partial x} \frac{dx}{dt} + \frac{\partial z}{\partial y} \frac{dy}{dt}$	DEPENDENCY DIAGRAM A GRAPHICAL REPRESENTATION OF THE CHAIN RULE.	THE GRADIENT VECTOR $\nabla f(x, y) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right)$
IMPACT DIFFERENTIATION $\frac{dz}{dx} = \frac{\partial z}{\partial x} + \frac{\partial z}{\partial y} \frac{dy}{dx}$	DIRECTIONAL DERIVATIVES $D_{\mathbf{u}} f(x, y) = \nabla f \cdot \mathbf{u}$	MAXIMUM AND MINIMUM VALUES $z = f(x, y)$ TO FIND THE LOCAL MAX. AND MIN. VALUES, SET $f_x = 0$ AND $f_y = 0$.
TANGENT PLANE TO A LEVEL SURFACE $f(x, y, z) = k$ $\nabla f(x_0, y_0, z_0) \cdot (x - x_0, y - y_0, z - z_0) = 0$	THE GRADIENT VECTOR IS ORTHOGONAL TO THE LEVEL CURVES OF A SURFACE	FINDING ABSOLUTE MAX. AND MINS. 1. Find values of f at all critical points of f in D . 2. Find the extreme values of f on the boundary of D . 3. The largest value from 1, 2 is the ABS. MAX. and the smallest value is the ABS. MIN.
NORMAL LINE TO A LEVEL SURFACE $\mathbf{n} = \nabla f(x_0, y_0, z_0)$	TO FIND THE NORMAL (AND TANGENT) LINE TO A SURFACE, LET THAT SURFACE BE THE LEVEL SET OF SOME NUMBER DIMENSIONAL FUNCTION. THEN THE GRADIENT OF THE FUNCTION AT POINT (a, b, c) IS A NORMAL VECTOR TO THE SURFACE AT (a, b, c).	

PHYSICS

VELOCITY
displacement / time
force / mass
momentum / mass
impulse / time
work / energy

ACCELERATION
change in velocity / time
force / mass

GRAVITATIONAL POTENTIAL
work done / mass
gravitational potential energy / mass

WAVES
transverse wave
longitudinal wave
standing wave
interference
diffraction

OPTICS
reflection
refraction
total internal reflection
dispersion

RELATIVITY
time dilation
length contraction
mass increase

ATMOSPHERE
spectral lines
refraction index
scattering

GRAVITY
gravitational field strength
gravitational potential
gravitational potential energy

ENERGY
kinetic energy
potential energy
work done
power

STATISTICS
mean
standard deviation
variance

MATHEMATICS
differentiation
integration
vectors

CONVERSIONS
m/s to km/h
km/h to m/s

DIAGRAMS
Free body diagrams
Force diagrams
Wave diagrams
Ray diagrams

DEFINITIONS
Scalar vs Vector
Displacement vs Distance
Speed vs Velocity
Acceleration vs Deceleration

FORMULAS
 $v = \frac{d}{t}$
 $a = \frac{dv}{dt}$
 $s = \frac{1}{2}at^2$
 $v^2 = u^2 + 2as$
 $F = ma$
 $E = mc^2$
 $E = hf$
 $n_1 \sin \theta_1 = n_2 \sin \theta_2$

EXPLANATIONS
Why does a body's speed remain constant if the net force acting on it is zero?
Why is acceleration directly proportional to net force and inversely proportional to mass?
How does gravity affect the path of a projectile?
Why does light bend when it passes from one medium to another?
How does the refractive index of a medium affect the speed of light?

EXAMPLES
A car starts from rest and accelerates at 2 m/s^2 for 5 s . Calculate its final velocity and the distance it travels.
A ball is thrown vertically upwards with an initial velocity of 10 m/s . Calculate the maximum height it reaches and the time it takes to return to the ground.
A light ray is incident on a boundary between air and water at an angle of 30° . Calculate the angle of refraction.

Admin

THE PURPOSE OF THIS REFERENCE GUIDE IS TO WALK THROUGH THE PROCESS OF BOOTING THE SIFT WORKSTATION, CREATING A TIMELINE ("SUPER" OR "MICRO") AND REVIEWING IT.

HOW TO CALCULATE THE OFFSET FOR MOUNTING
1. Run nmls to query partition layout
nmls image.E01
2. Identify partition and byte offset
3. (Partition byte offset) x (bytes per sector) = offset ##### to use!
Example: 63 x 512 = 32256

Note: if needed, repeat for each partition. Make new mount point: # mkdir /mnt/windows_mount2/

- log2timeline PARSING PLUGINS
apach2_error - Apache2 error log file
chrome - Chrome history file
emcaac_drfiling - CSV file that is exported from emcaac
event - Windows %SystemRoot%\System32\Winevt\Logs\event - Windows Event Log file (EVTX)
fsinfo - Metadata information from files using TeFileTool
fx_bookmark - Firefox bookmark file
fx_cookies - Firefox 2 browser history file
fx_cookies2 - Firefox 3 history file
fx_cookies3 - CSV file that is exported from Firefox (Firefox)
generic_logs - Generic (user logs that start with MM/DD/YYYY)
history - Individual file containing Firefox history
ls - LS WSC log file
lsat - LS text export log file
ls_nfs_change - CSV output file from SIFT [NTFS Change log]
machine - Body file in the machine format
mcafee - Log file
nfs - NFS MFT file
nmls_emlog - ERRORLOG file produced by MySQL server
ntuser.dat_registry - NTUSER.DAT registry file
opera - Opera's global history file
osint - (optional) document prep
pcap - PCAP file
pdf - Available PDF document metadata
prefetch - Prefetch directory
recycle - Recycle bin directory
restore - Restore point directory
safari - Safari History.plist file
sam - SAM registry file
security - SECURITY registry file
setupapi - SetupAPI log file in Windows XP
shape_sql - Shape database software - SOFTWARE registry file
sql - .sql/.SQL or a Flash cookie file
sqlid - SQLid access log (http://www.sqlid.org)
spring - Spring log file
system - SYSTEM registry file
tfs - Body file in the TFS format
volatility - Volatility output files (process2, services2, ...)
win_log - Windows shortcut file for a link file
windows - windows log file
xpsetup - XP Firewall log

BY DAVID NIDES (12/24/2012)
TWITTER: @DANNIDES
BLOG: DANNIDES.BLOGSPOT.COM
EMAIL: DANNIDES@PRAGMATIC.COM
CREDITS FOR: ED GOINGS, ROB LEWIS, KRISTIAN GUSHONKES, KPRAGMATIC.COM
QUESTIONS/FEEDBACK-CONTACT US

KEY
Red text - Image/Source
Blue text - Mount point
Purple text - output file
Green text - log2timeline plugins
Brown text - Timezone



Table with 5 columns: File System, M, A, C, E. Rows include ext2/3, FAT, NTFS, and UFS.

HELP? OPTIONS? USAGE?
log2timeline -help
log2timeline-oft -help
L2L_process -help

OTHER log2timeline OUTPUT FORMATS
Note: CSV is Default Output
-CSV - comma separated value file
-JSON - Both older and newer version of the format supported for use by SIFT's machine
-XML - XML file - SMALE timeline visualization widget
-JSON - JSON database
-JSON - Tab Delimited File
-JSON - Format used by some of H Carvey tools, expressed as an ASCII output
-XML - Format used by some of H Carvey tools, expressed as a XML document

- 10. CONNECT TO SIFT
1. SIFT SETTINGS -> OPTIONS -> Shared Settings -> Always Enabled [Check]
2. SIFT Desktop -> VMware-Shared-Drive
Access from a Win Machine
\SIFTWORKSTATION

11. REVIEW TIMELINE
Review timelines using:
- Open, Soft, Filter with Excel
- Import into SPLUNK
- SMALE
- Tapestry

CS 110
Computer Architecture
Lecture 30:
Course Summary
Video 2: WSC to CPUs

Instructors:

Sören Schwertfeger & Chundong Wang

<https://robotics.shanghaitech.edu.cn/courses/ca/20s/>

School of Information Science and Technology SIST

ShanghaiTech University

Slides based on UC Berkley's CS61C

New School Computer Architecture (1/3)



Personal
Mobile
Devices

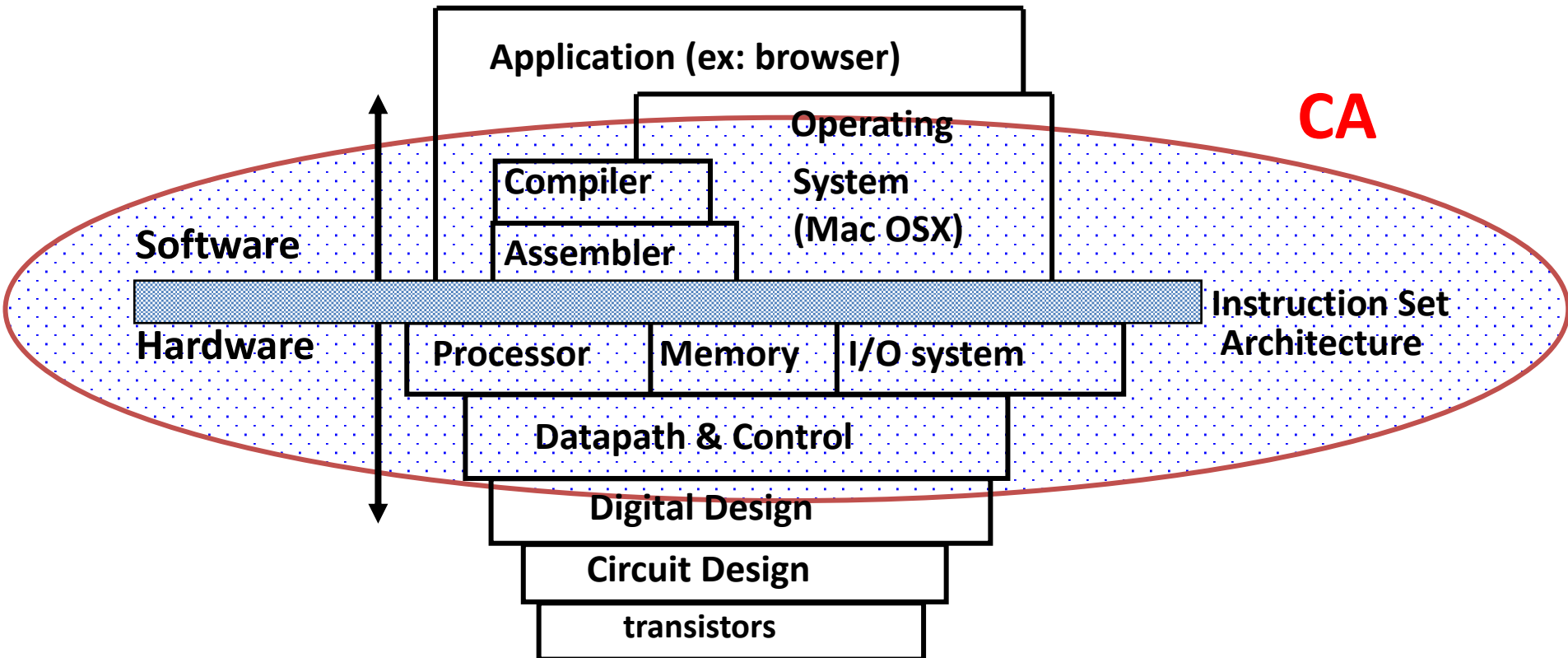
New School Computer Architecture (2/3)



New School Computer Architecture (3/3)



Old Machine Structures



New-School Machine Structures (It's a bit more complicated!)

Software

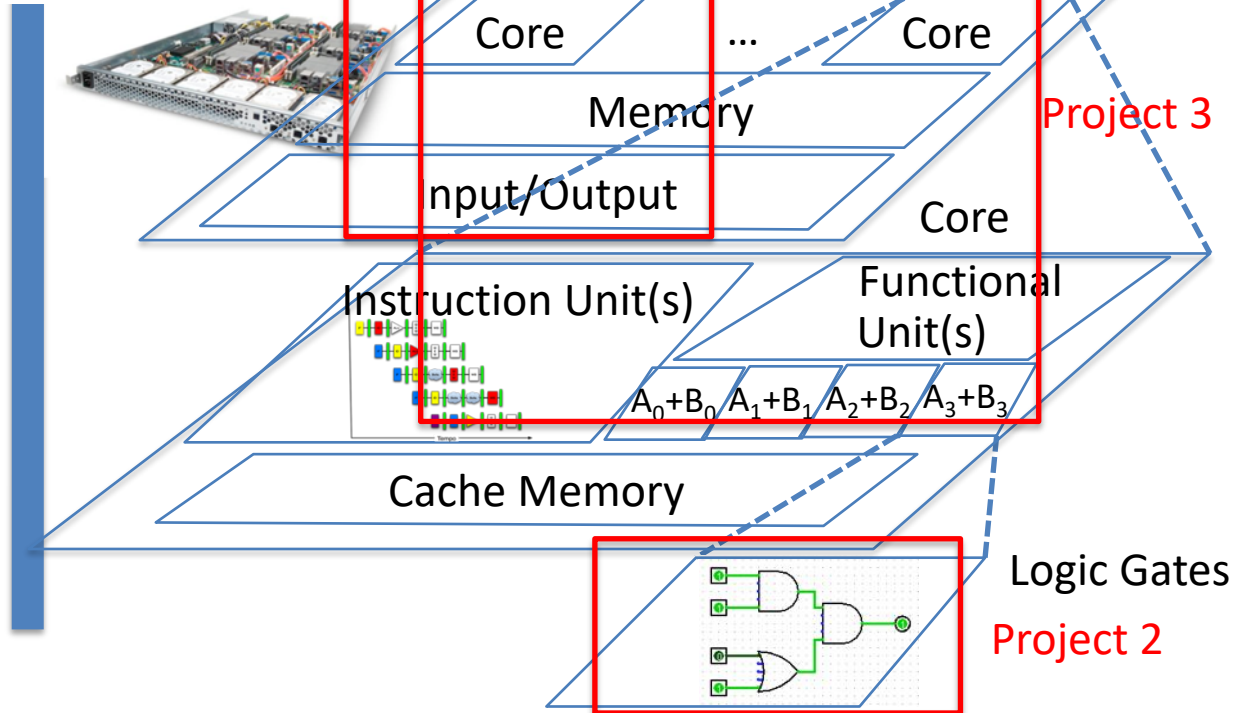
Hardware

Warehouse
Scale
Computer

Smart
Phone



*Leverage
Parallelism &
Achieve High
Performance*



- Parallel Requests**

Assigned to computer
e.g., Search “Katz”

- Parallel Threads**

Assigned to core
e.g., Lookup, Ads

- Parallel Instructions**

>1 instruction @ one time
e.g., 5 pipelined instructions

- Parallel Data**

>1 data item @ one time
e.g., Add of 4 pairs of words

- Hardware descriptions**

All gates functioning in
parallel at same time

- Programming Languages**

Great Ideas in Computer Architecture

1. Design for Moore's Law
2. Abstraction to Simplify Design
3. Make the Common Case Fast
4. Dependability via Redundancy
5. Memory Hierarchy
6. Performance via
Parallelism/Pipelining/Prediction

Powers of Ten inspired CA Overview

- Going Top Down cover 3 Views
 1. Architecture (when possible)
 2. Physical Implementation of that architecture
 3. Programming system for that architecture and implementation (when possible)
- See <http://www.powersof10.com/film>

Earth

10^7 meters

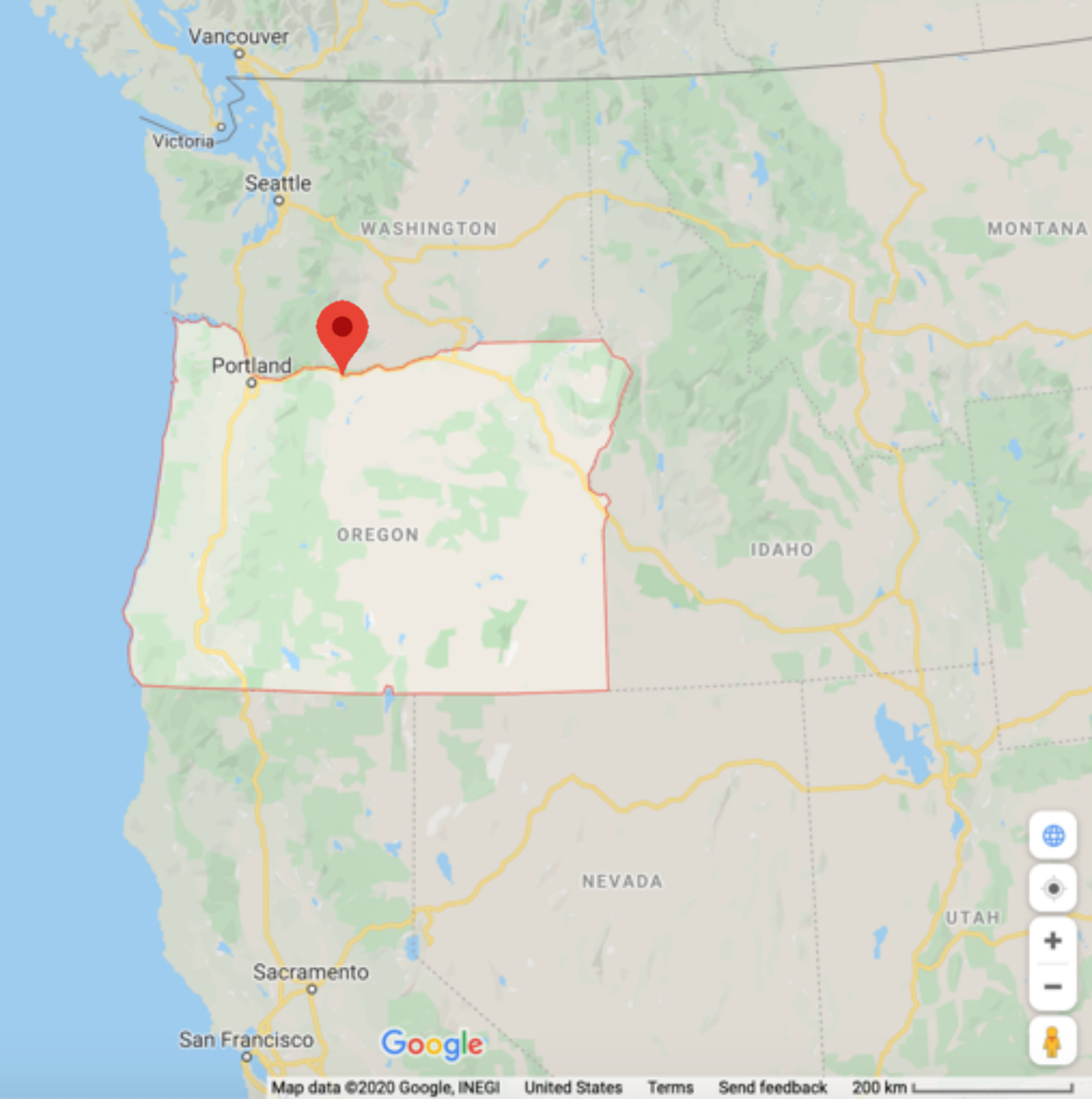


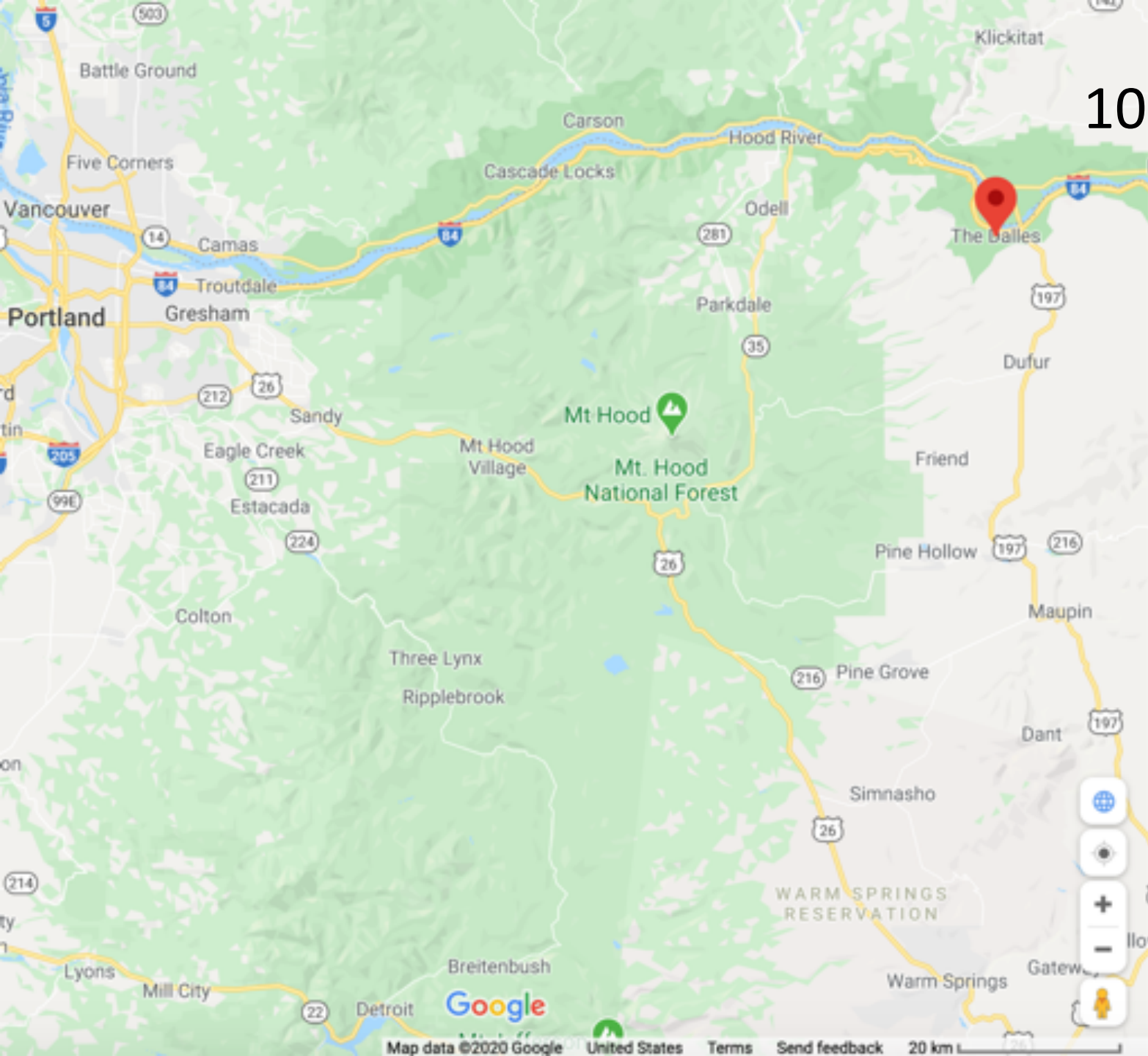
10^7 meters



Google

10^6 meters





10^5 meters

The Dalles, Oregon 10^4 meters



The Dalles, Oregon 10^4 meters



Google's Oregon WSC 10^3 meters



10^4 meters

Google's Oregon WSC

10 kilometers



10^2 meters



10^3 meters



Google Warehouse

- 90 meters by 75 meters, 10 Megawatts
- Contains 40,000 servers, 190,000 disks
- Power Utilization Effectiveness: 1.23
 - 85% of 0.23 overhead goes to cooling losses
 - 15% of 0.23 overhead goes to power losses
- Contains 45, 40-foot long containers
 - 8 feet x 9.5 feet x 40 feet
- 30 stacked as double layer, 15 as single layer

Containers in WSCs

10² meters



100 meters

Google Container

10¹ meters

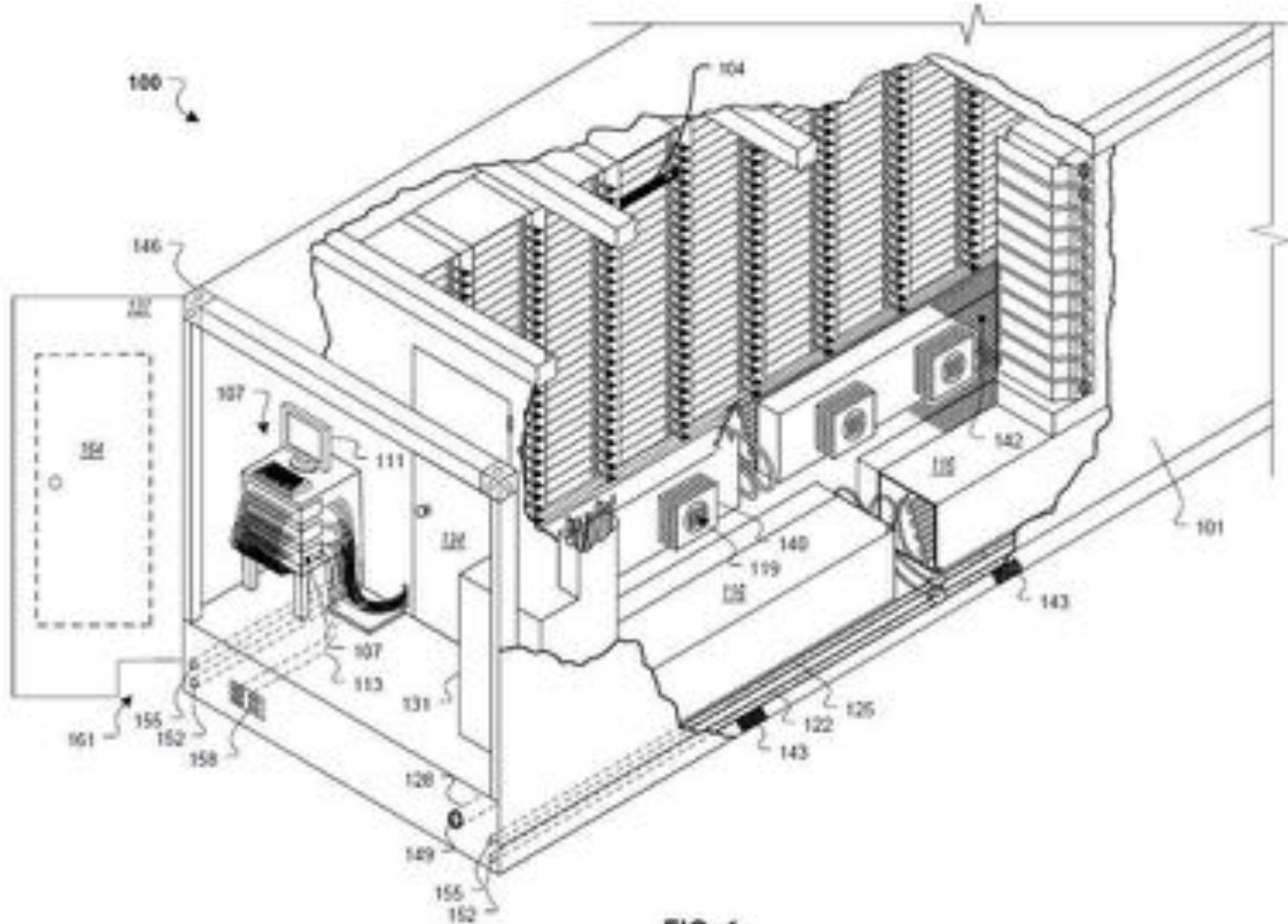
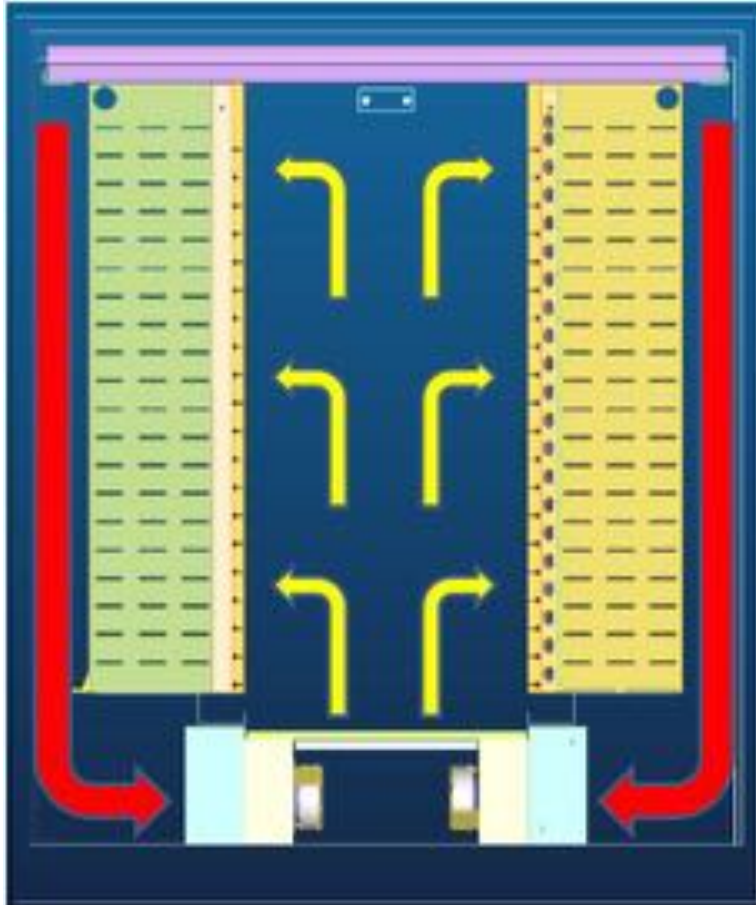


FIG. 1

Google Container 10^0 meters

10 meters



- 2 long rows, each with 29 racks
- Cooling below raised floor
- Hot air returned behind racks

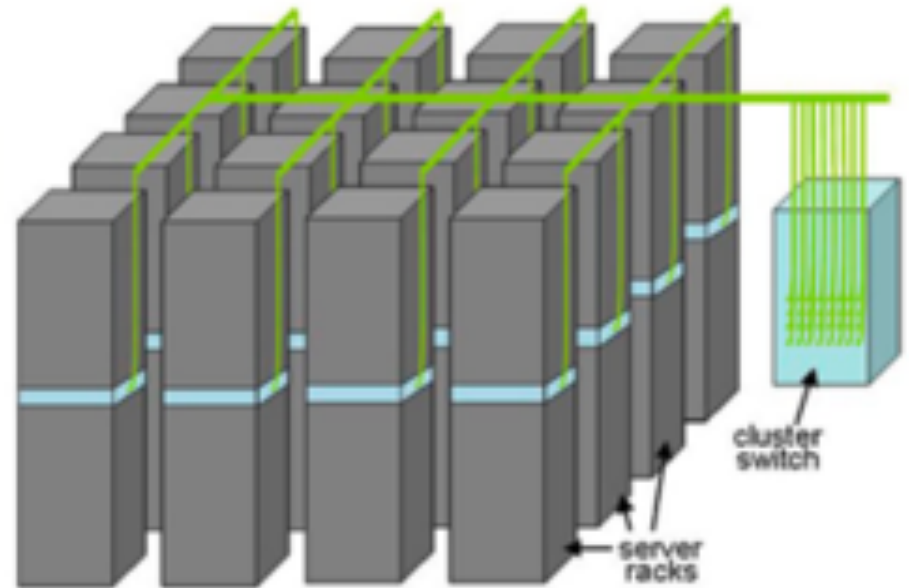
Equipment Inside a Container



Server (in rack format):



7 foot Rack: servers + Ethernet local area network switch in middle (“rack switch”)



Array (aka cluster):
server racks + larger local area network switch (“array switch”) 10X faster => cost 100X: cost $f(N^2)$

Great Ideas in Computer Architecture

1. *Design for Moore's Law*
 - *WSC, Container, Rack*
2. Abstraction to Simplify Design
3. Make the Common Case Fast
4. *Dependability via Redundancy*
 - *Multiple WSCs, Multiple Racks, Multiple Switches*
5. Memory Hierarchy
6. *Performance via Parallelism/Pipelining/Prediction*
 - *Task level Parallelism, Data Level Parallelism*

Google Server Internals 10^{-1} meters

10 centimeters



Facebook Datacenter

facebook



Sfotware: Often uses MapReduce

- Simple data-parallel ***programming model*** and ***implementation*** for processing large datasets
- Users specify the computation in terms of
 - a ***map*** function, and
 - a ***reduce*** function
- Underlying runtime system
 - Automatically ***parallelize*** the computation across large scale clusters of machines
 - ***Handles*** machine ***failure***
 - ***Schedule*** inter-machine communication to make efficient use of the networks

Programming Multicore Microprocessor: OpenMP

```
#include <omp.h>
#include <stdio.h>
static long num_steps = 100000;
int value[num_steps];
int reduce()
{   int i;   int sum = 0;
#pragma omp parallel for private(x) reduction(+:sum)
    for (i=1; i<= num_steps; i++){
        sum = sum + value[i];
    }
}
```


Great Ideas in Computer Architecture

1. *Design for Moore's Law*
 - *More transistors = Multicore + SIMD*
2. Abstraction to Simplify Design
3. Make the Common Case Fast
4. Dependability via Redundancy
5. *Memory Hierarchy*
 - *More transistors = Cache Memories*
6. *Performance via Parallelism/Pipelining/Prediction*
 - *Thread-level Parallelism*

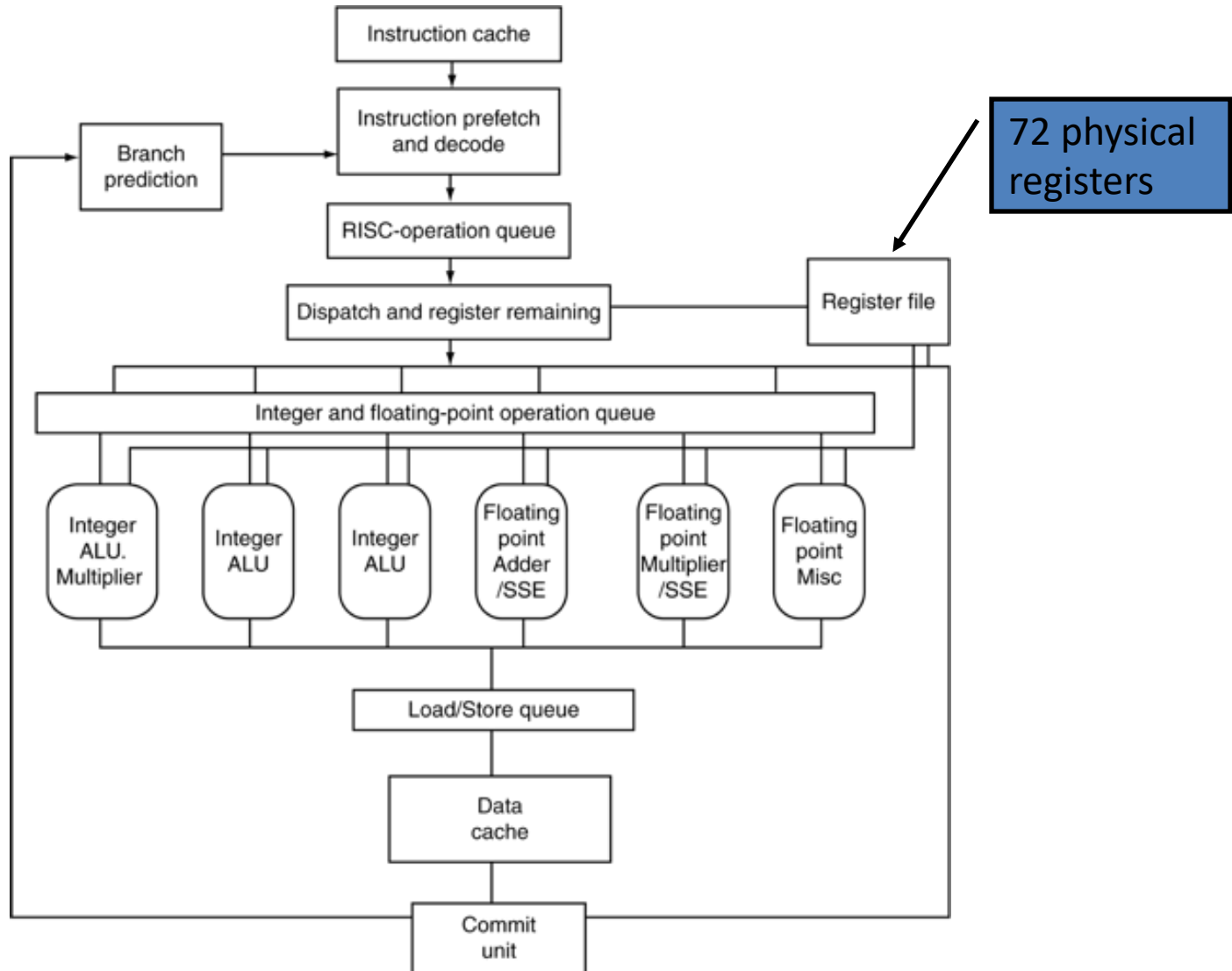
10^{-2} meters

AMD Opteron Microprocessor

centimeters

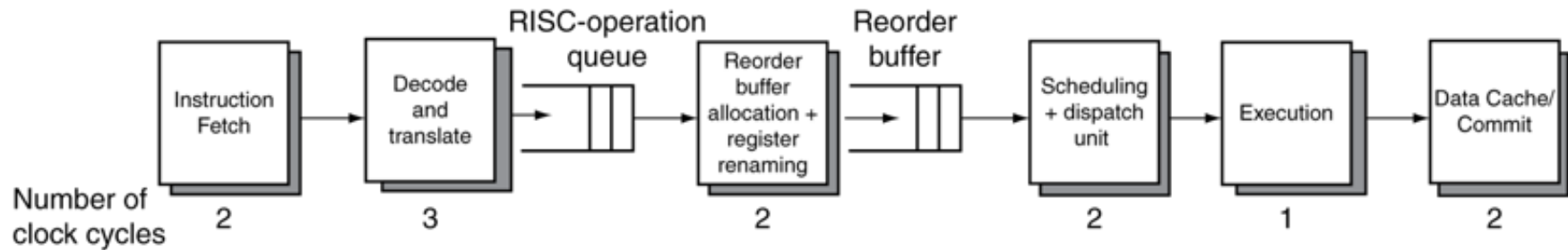


AMD Opteron Microarchitecture



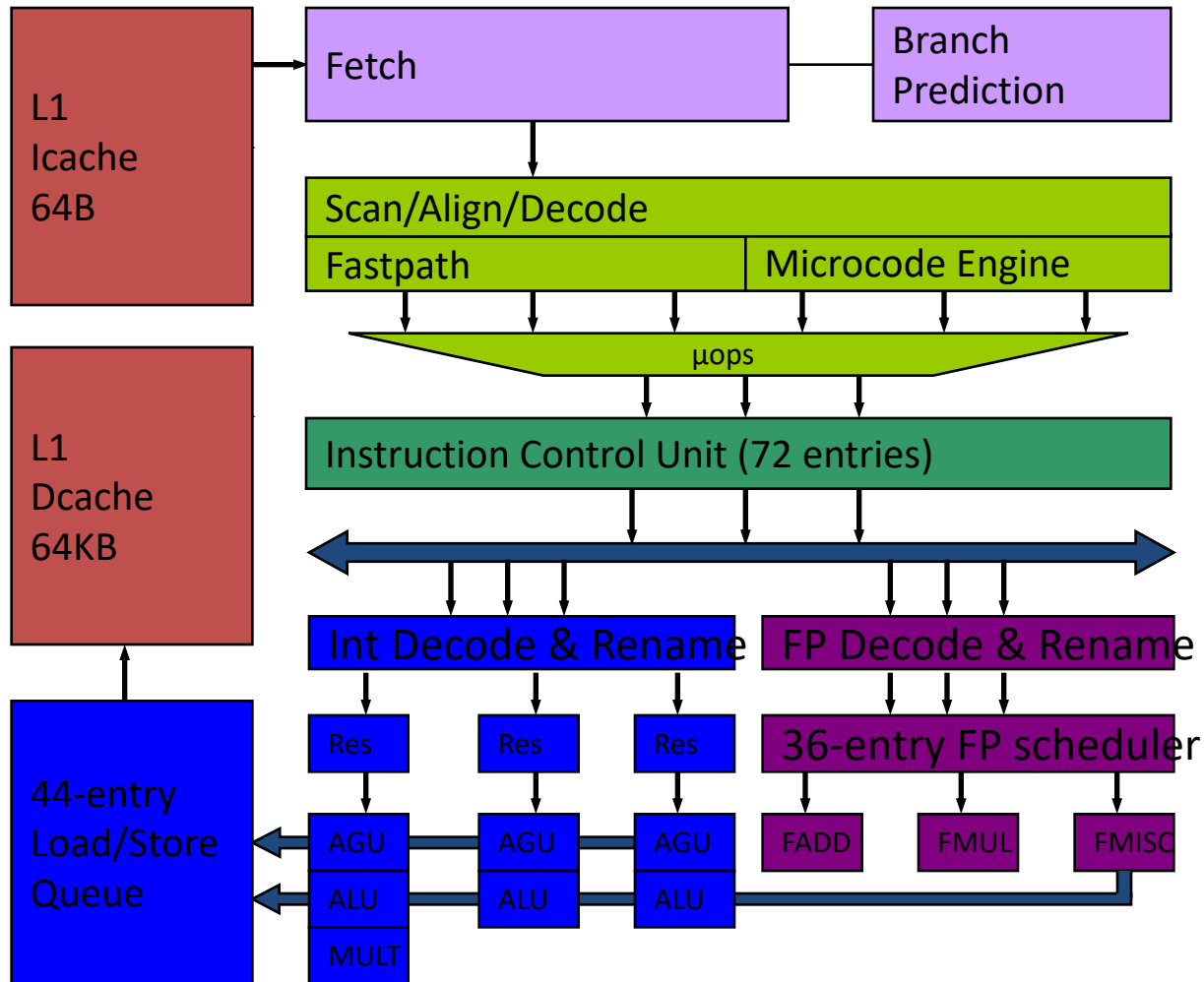
AMD Opteron Pipeline Flow

- For integer operations



- 12 stages (Floating Point is 17 stages)
- Up to 106 RISC-ops in progress

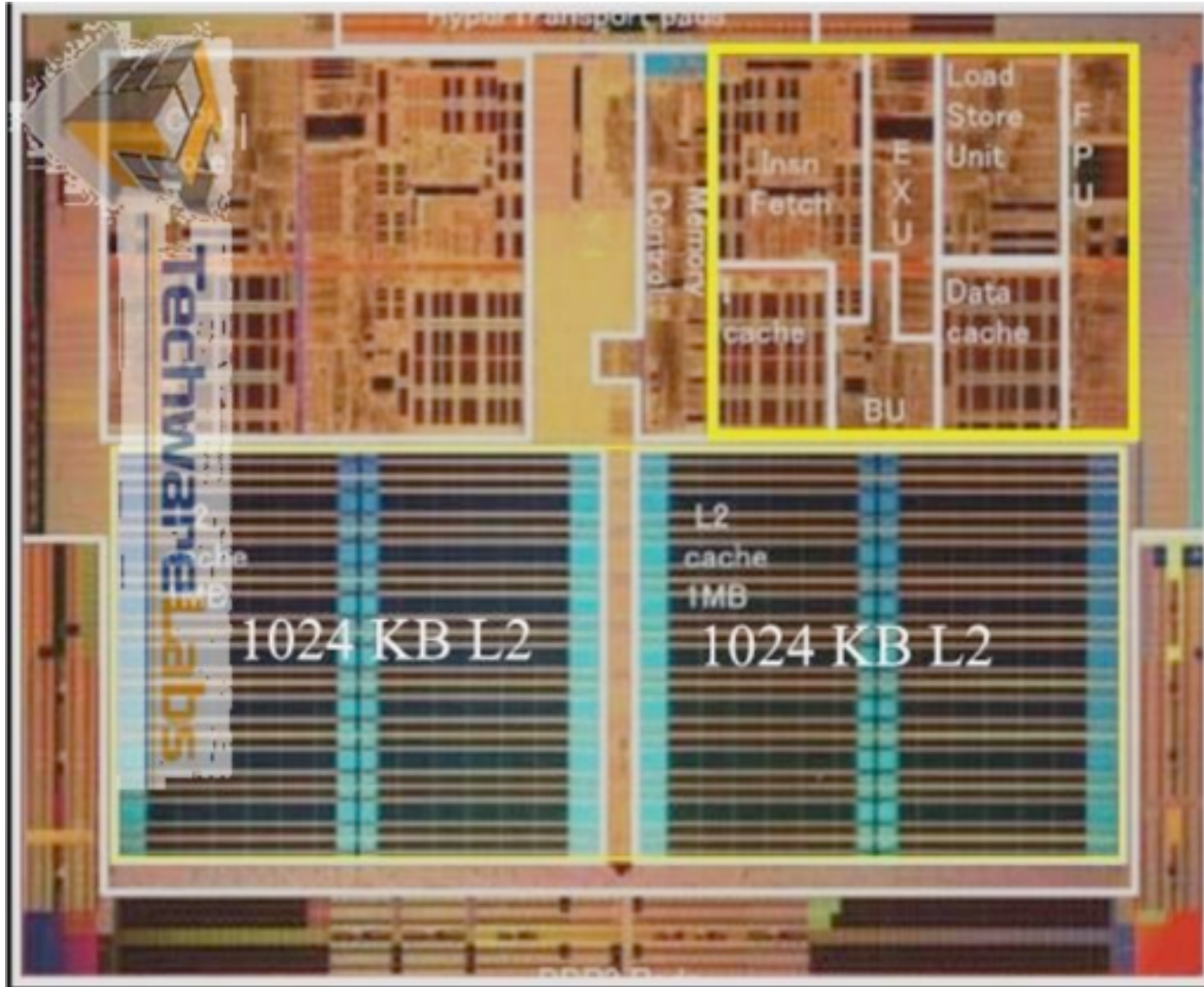
AMD Opteron Block Diagram



10⁻² meters

AMD Opteron Microprocessor

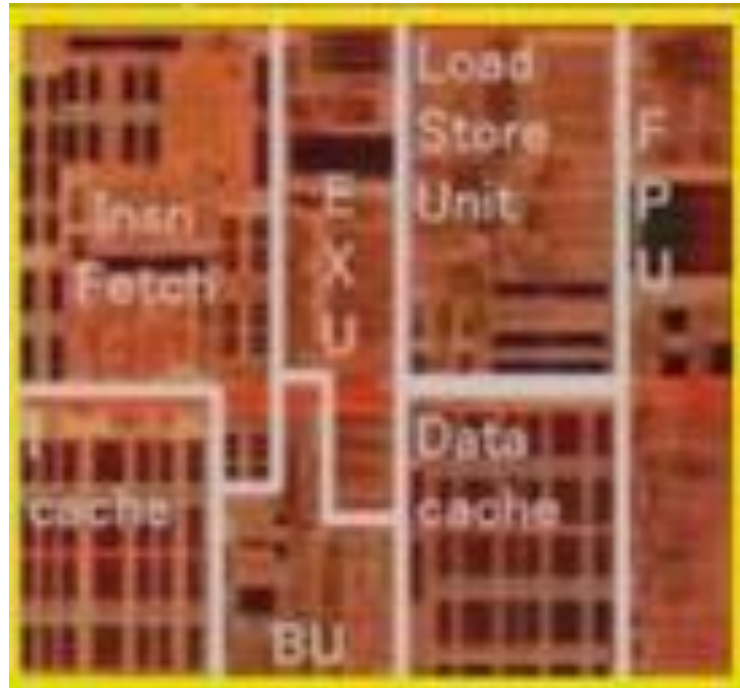
centimeters



10⁻³ meters

AMD Opteron Core

millimeters



Zoom into a Microchip

Zoom into a Microchip

From a Digital Camera
To a Scanning Electron
Microscope

Produced by
NISE Net



Q & A



Remember:

- One more Video ;)
- Prepare Questions for next session

CS 110
Computer Architecture
Lecture 30:
Course Summary
Video 3: CPUs to Transistors

Instructors:

Sören Schwertfeger & Chundong Wang

<https://robotics.shanghaitech.edu.cn/courses/ca/20s/>

School of Information Science and Technology SIST

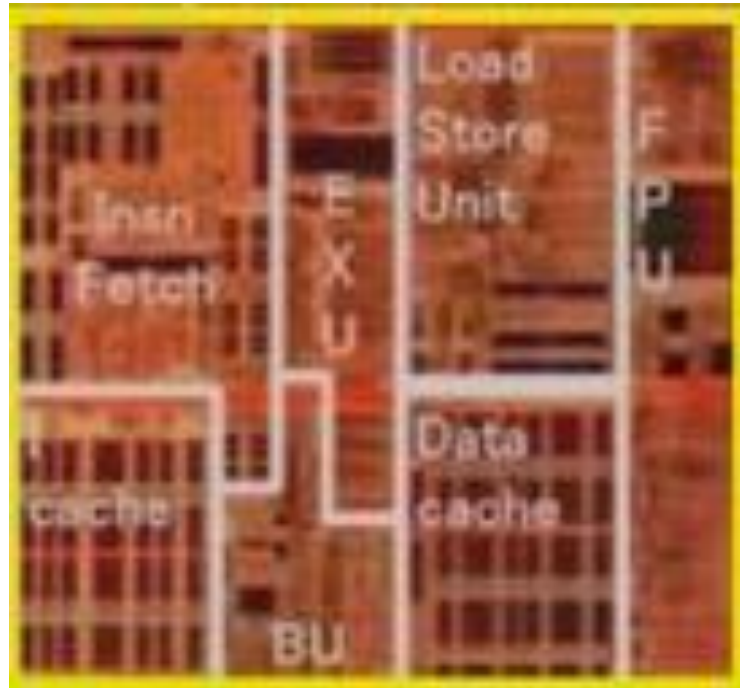
ShanghaiTech University

Slides based on UC Berkley's CS61C

10⁻³ meters

AMD Opteron Core

millimeters



Programming One Core: C with Intrinsics

```
void mmult(int n, float *A, float *B, float *C)
{
    for ( int i = 0; i < n; i+=4 )
        for ( int j = 0; j < n; j++ )
            {
                __m128 c0 = _mm_load_ps(C+i+j*n);
                for( int k = 0; k < n; k++ )
                    c0 = _mm_add_ps(c0, _mm_mul_ps(_mm_load_ps(A+i+k*n),
                                                    _mm_load1_ps(B+k+j*n)));
                _mm_store_ps(C+i+j*n, c0);
            }
}
```

Inner loop from gcc -O -S

Assembly snippet from innermost loop:

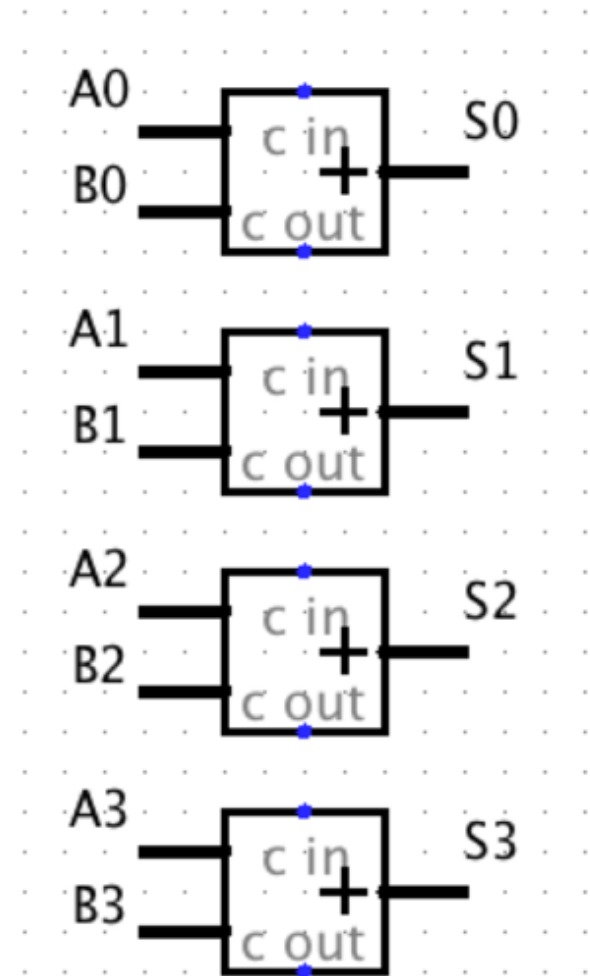
```
movaps (%rax), %xmm9
mulps  %xmm0, %xmm9
addps  %xmm9, %xmm8
movaps 16(%rax), %xmm9
mulps  %xmm0, %xmm9
addps  %xmm9, %xmm7
movaps 32(%rax), %xmm9
mulps  %xmm0, %xmm9
addps  %xmm9, %xmm6
movaps 48(%rax), %xmm9
mulps  %xmm0, %xmm9
addps  %xmm9, %xmm5
```

Great Ideas in Computer Architecture

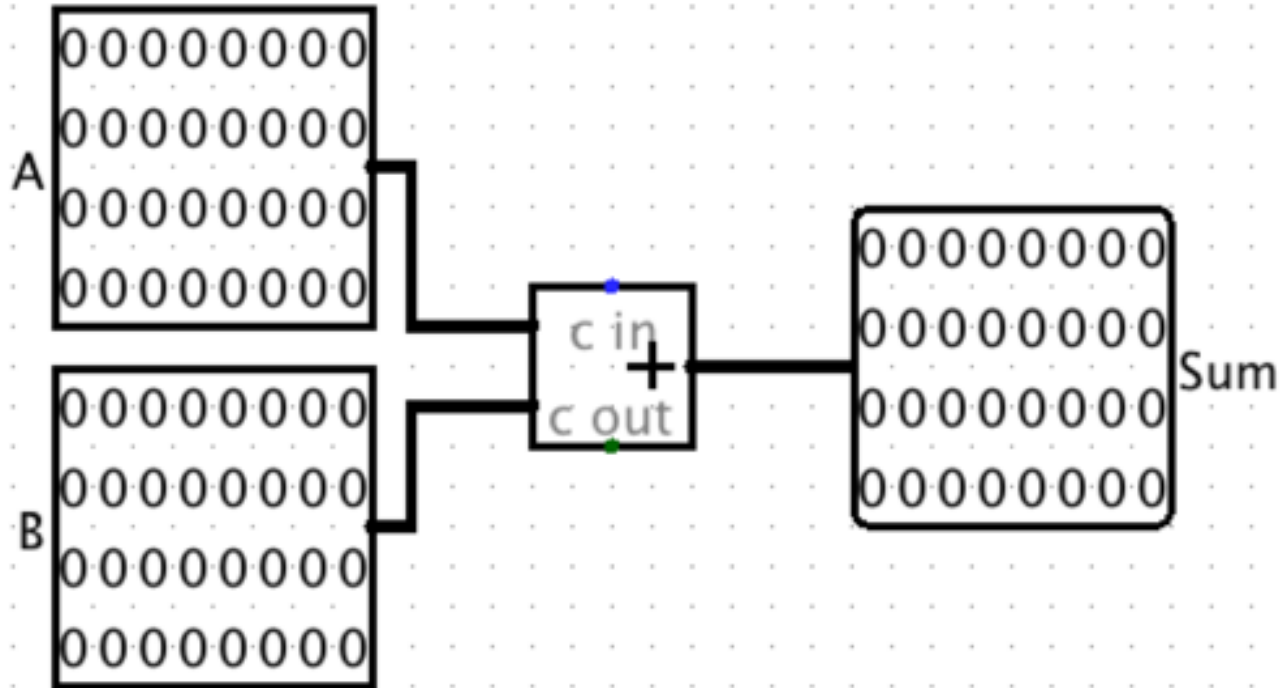
1. Design for Moore's Law
2. *Abstraction to Simplify Design*
 - *Instruction Set Architecture, Micro-operations*
3. Make the Common Case Fast
4. Dependability via Redundancy
5. Memory Hierarchy
6. *Performance via Parallelism/Pipelining/Prediction*
 - *Instruction-level Parallelism (superscalar, pipelining)*
 - *Data-level Parallelism*

SIMD Adder

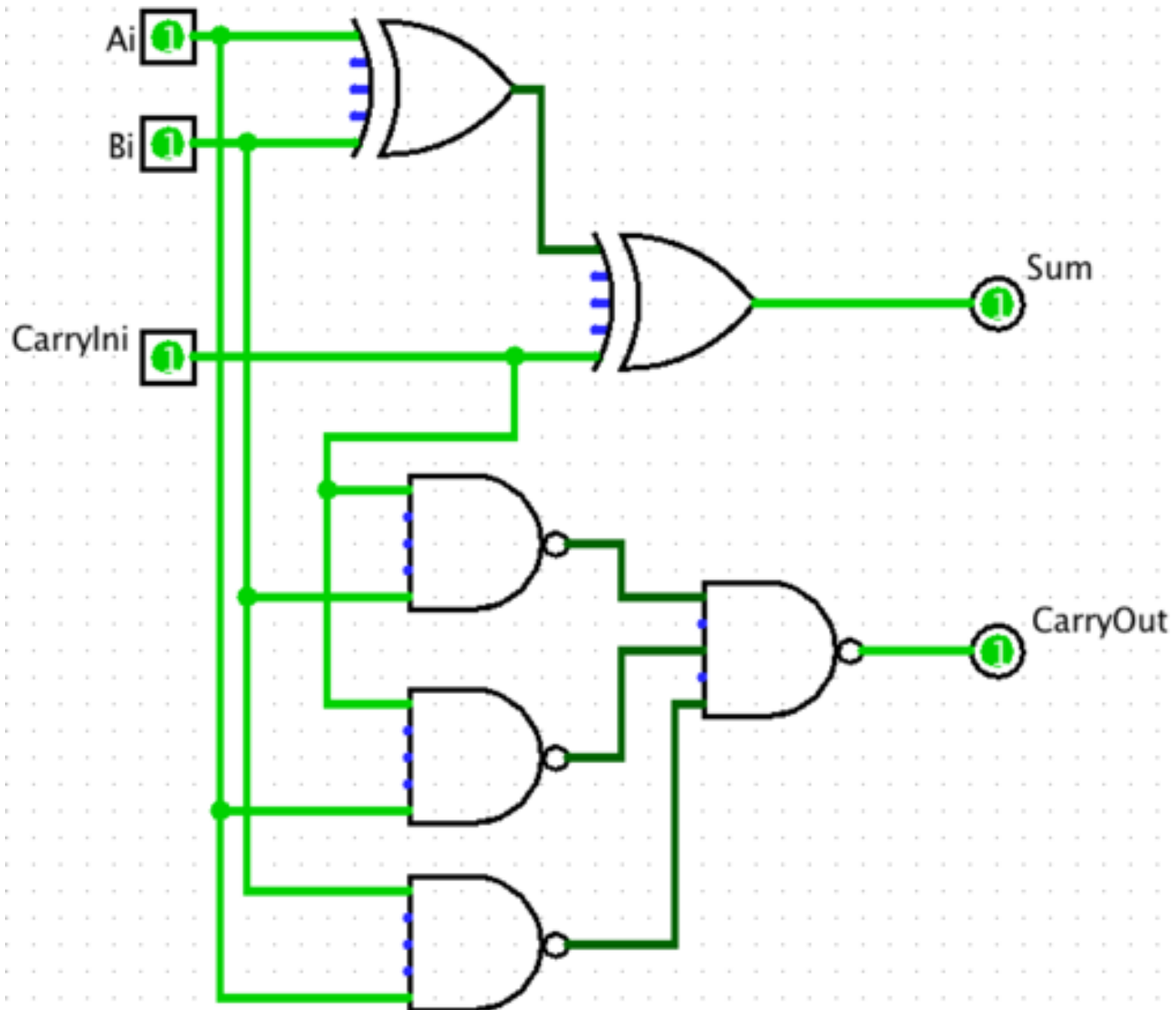
- Four 32-bit adders that operate in parallel
 - Data Level Parallelism



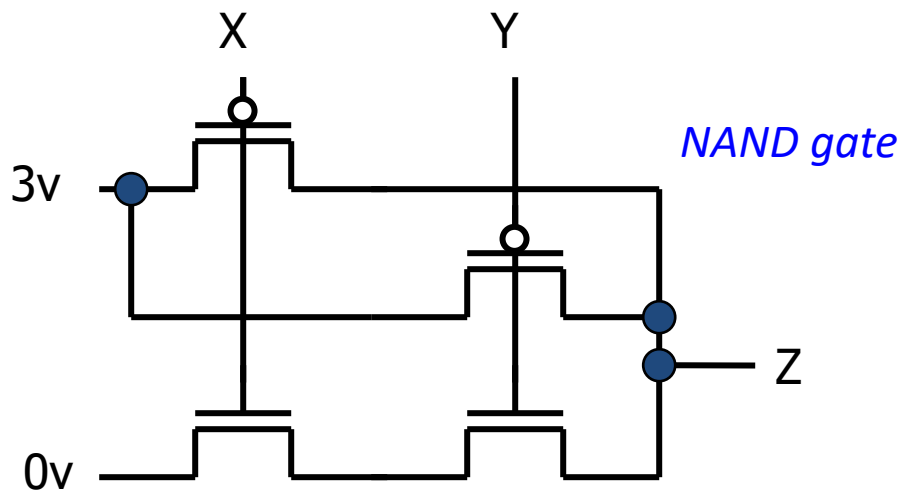
One 32-bit Adder



1 bit of 32-bit Adder



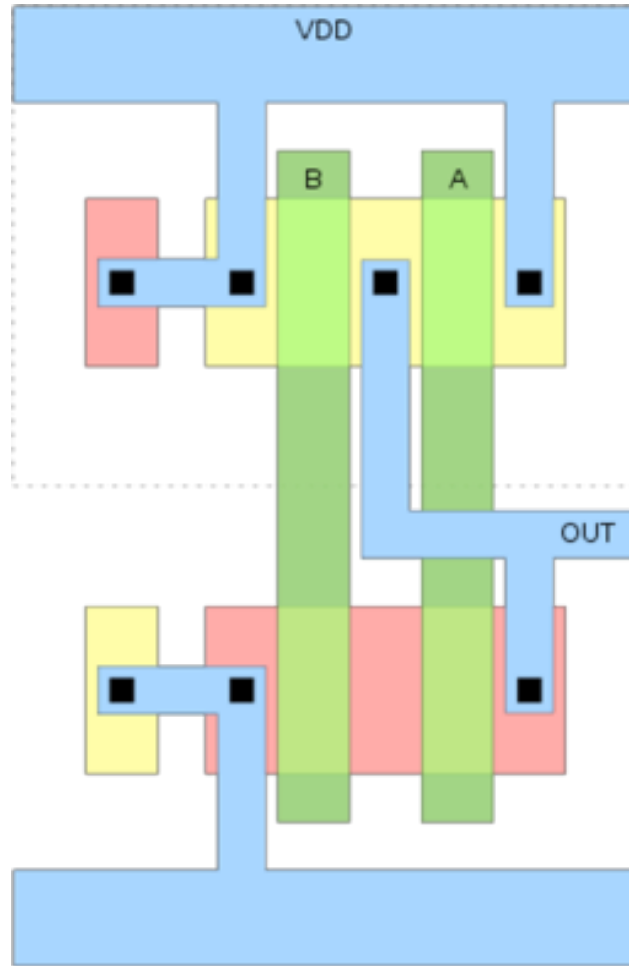
Complementary MOS Transistors (NMOS and PMOS) of NAND Gate



x	y	z
0 volts	0 volts	3 volts
0 volts	3 volts	3 volts
3 volts	0 volts	3 volts
3 volts	3 volts	0 volts

Physical Layout of NAND Gate 10^{-7} meters

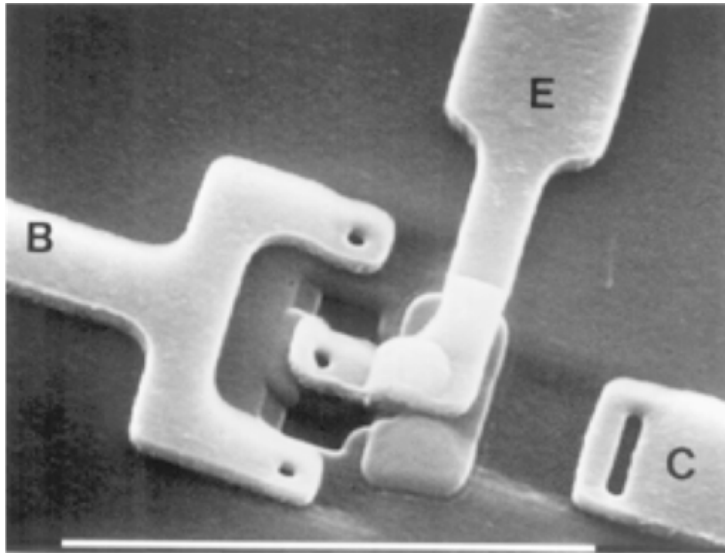
100 nanometers



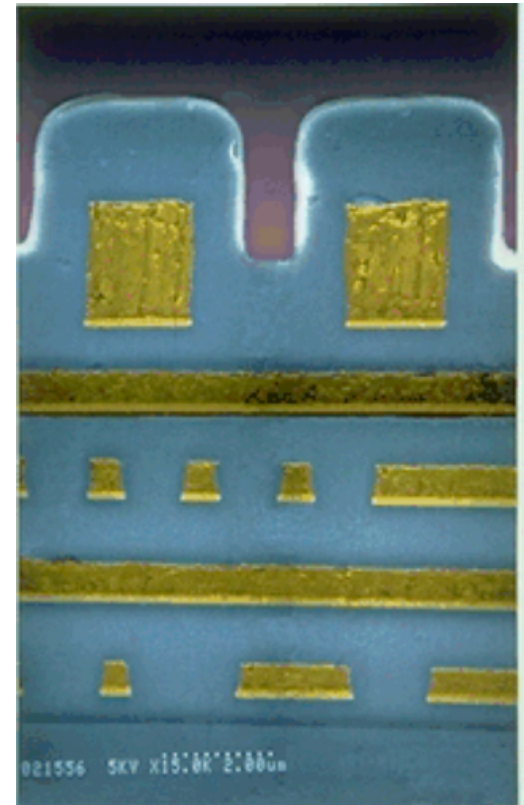
10^{-7} meters

Scanning Electron Microscope

100 nanometers



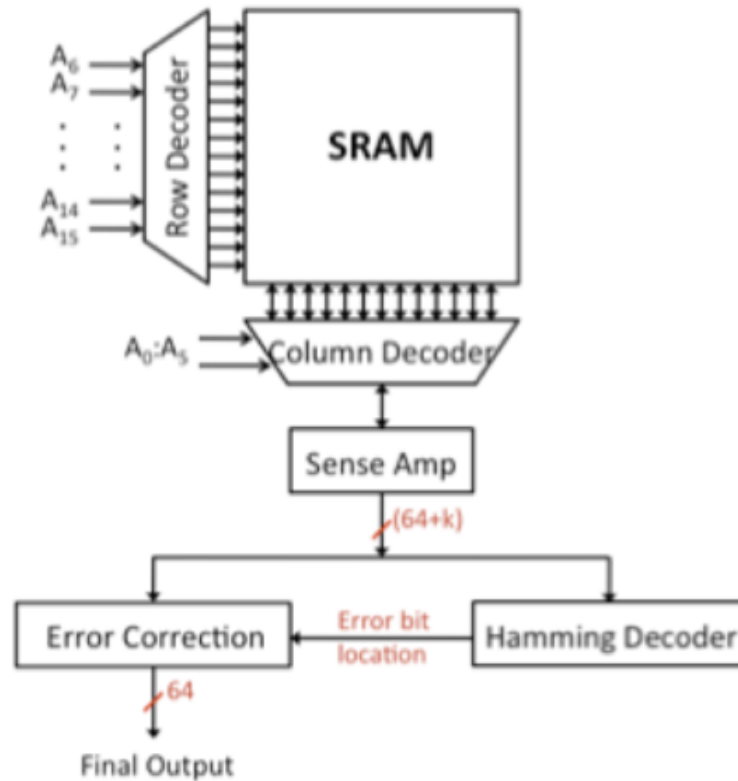
Top View



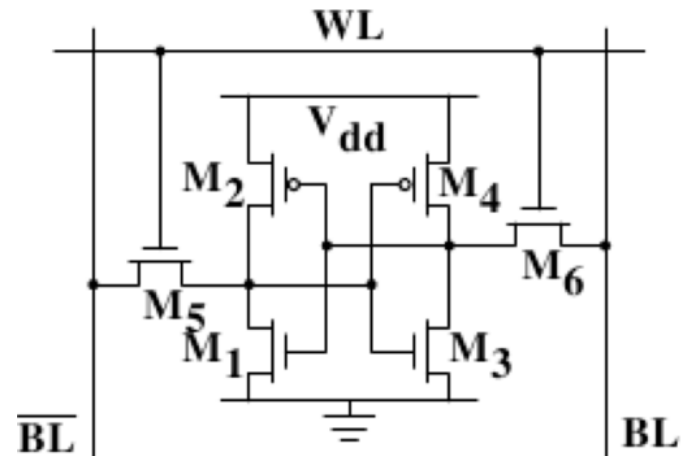
Cross Section

How to make a CMOS chip?

Block Diagram of Static RAM



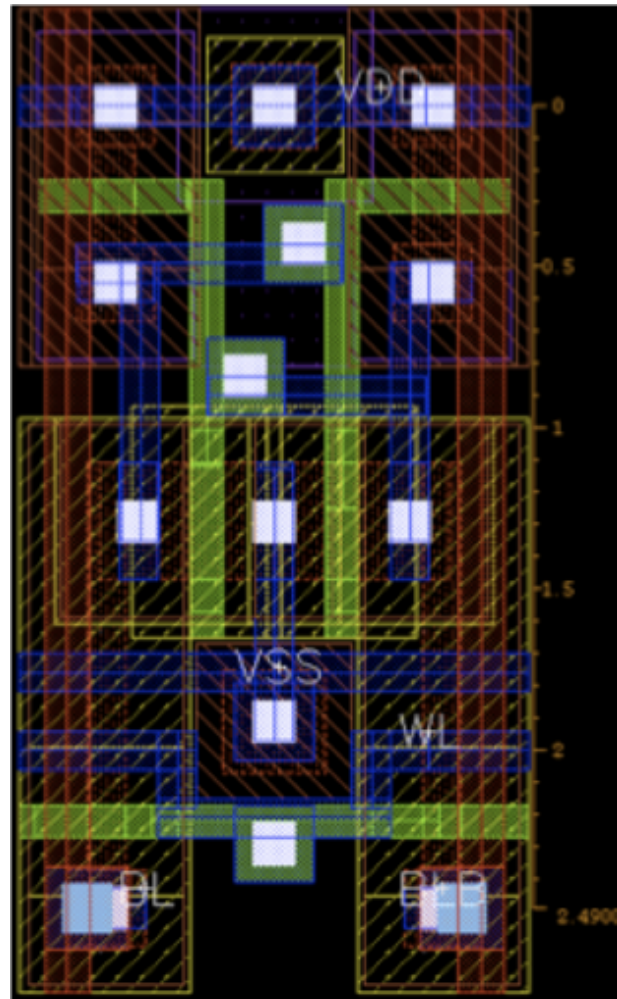
1 Bit SRAM in 6 Transistors



10^{-7} meters

Physical Layout of SRAM Bit

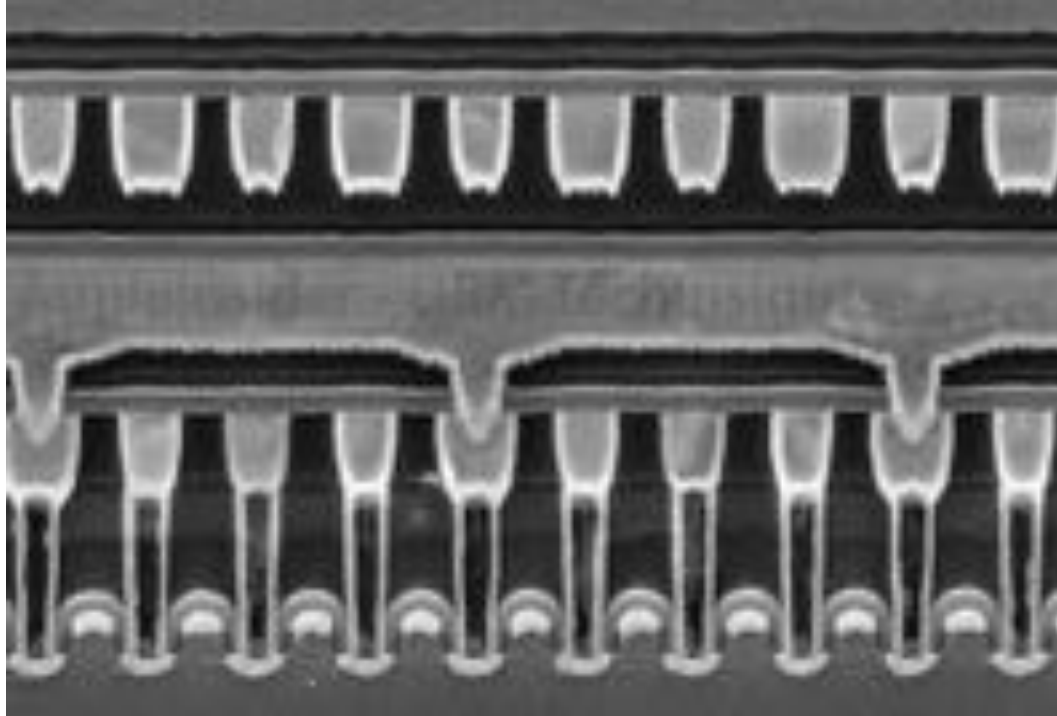
100 nanometers



10^{-7} meters

SRAM Cross Section

100 nanometers



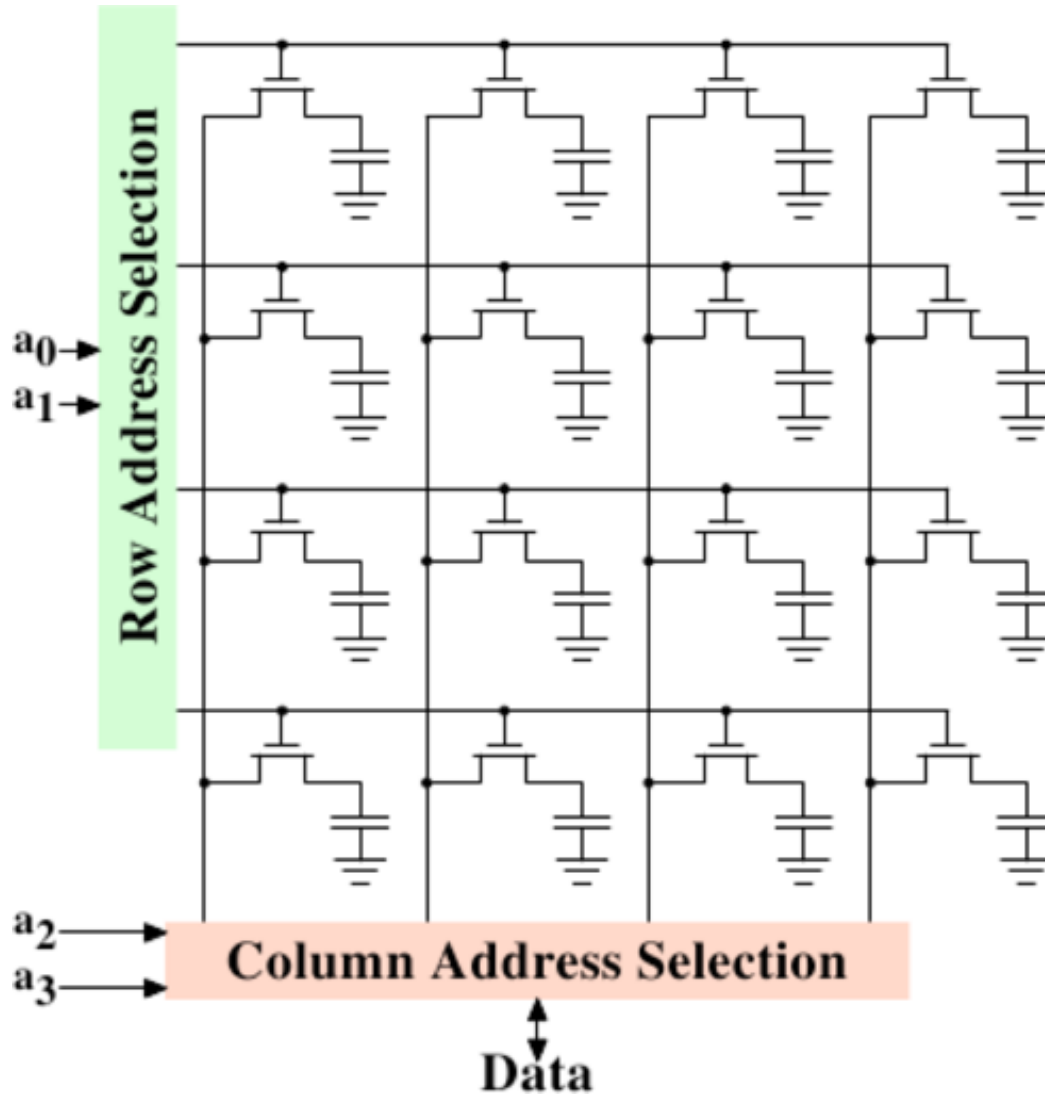
DIMM Module

- DDR = Double Data Rate
 - Transfers bits on Falling AND Rising Clock Edge
- Has Single Error Correcting, Double Error Detecting Redundancy (SEC/DED)
 - 72 bits to store 64 bits of data
 - Uses “Chip kill” organization so that if single DRAM chip fails can still detect failure
- Average server has 22,000 correctable errors and 1 uncorrectable error per year

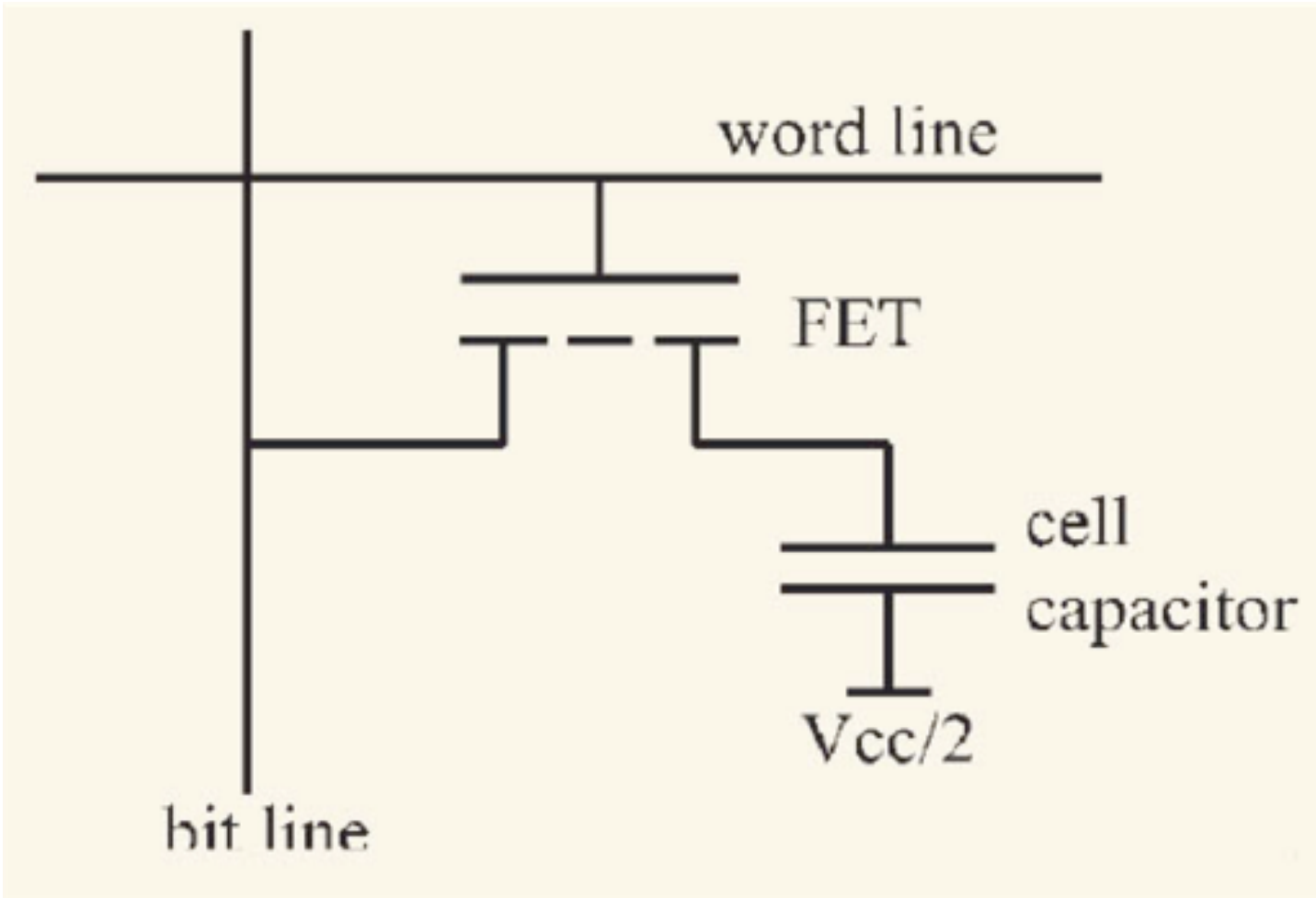
10^{-6} meters

DRAM Bits

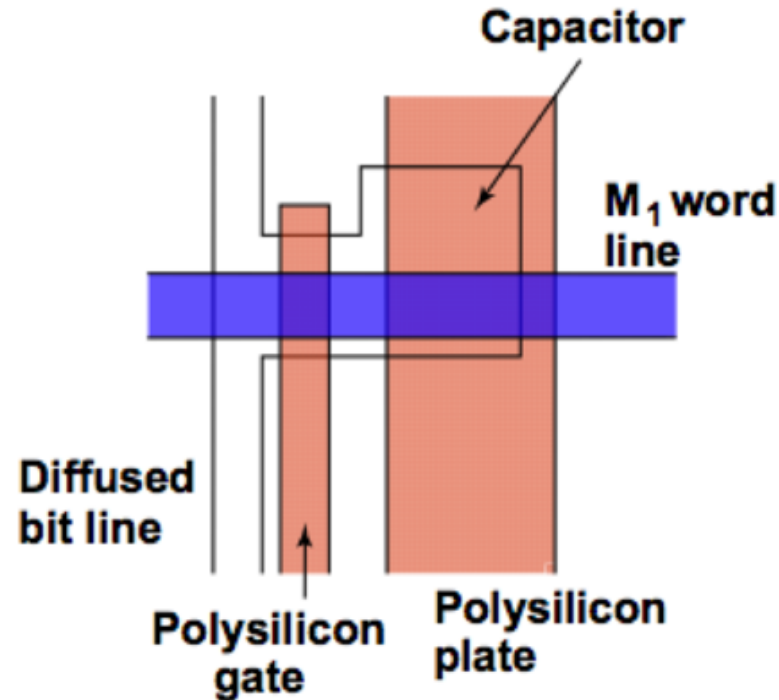
1 micron



DRAM Cell in Transistors



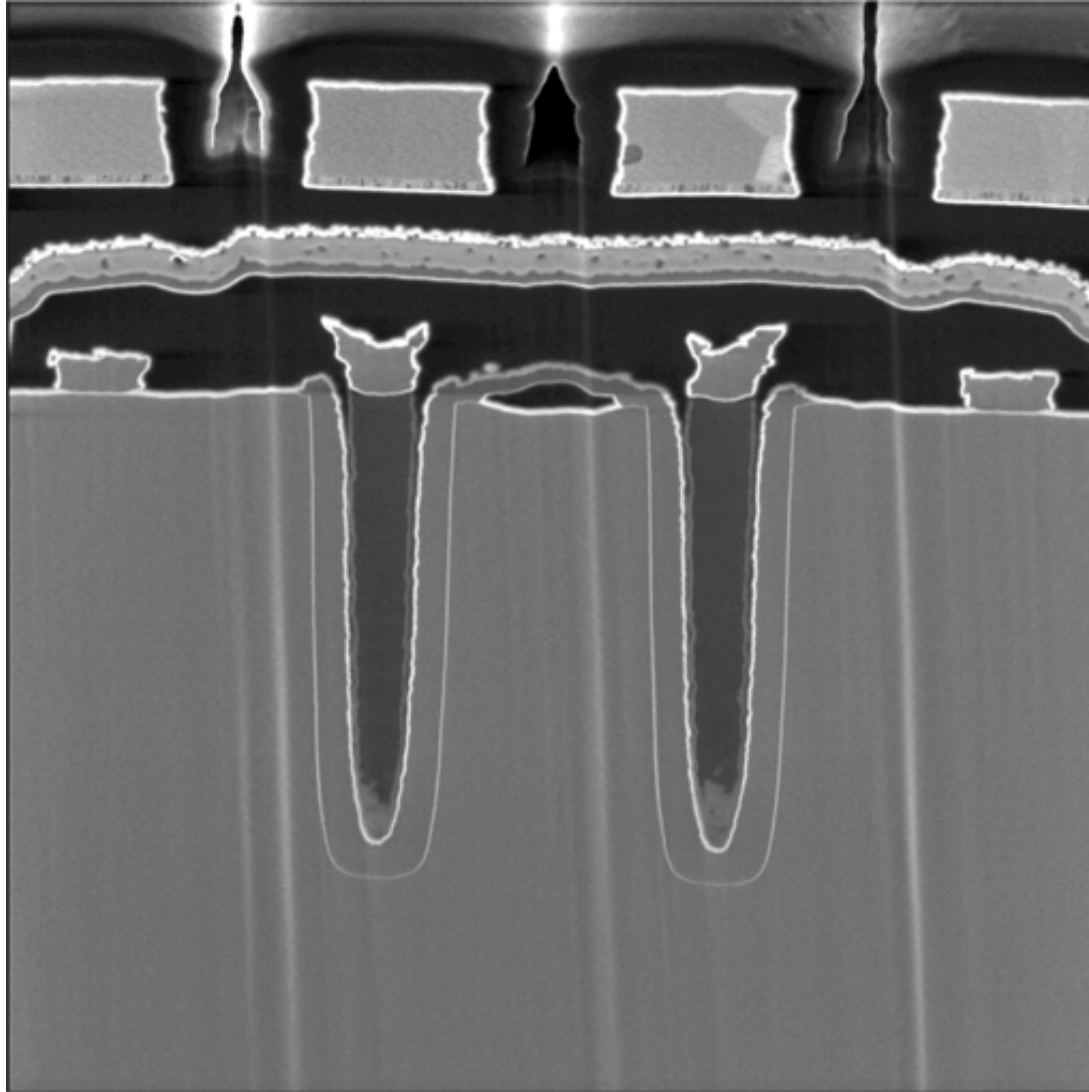
Physical Layout of DRAM Bit



10^{-7} meters

Cross Section of DRAM Bits

100 nanometers



TSCM



TSMC received 2,472 US patents,
increasing TSMC's total US patents to
19,336 (2018)

AMD Opteron Dependability

- L1 cache data is SEC/DED protected
- L2 cache and tags are SEC/DED protected
- DRAM is SEC/DED protected with chipkill
- On-chip and off-chip ECC protected arrays include autonomous, background hardware scrubbers
- Remaining arrays are parity protected
 - Instruction cache, tags and TLBs
 - Data tags and TLBs
 - Generally read only data that can be recovered from lower levels

Programming Memory Hierarchy: Cache Blocked Algorithm

- The blocked version of the i-j-k algorithm is written simply as (A,B,C are submatrices of a, b, c)

```
for (i=0;i<N/r;i++)
  for (j=0;j<N/r;j++)
    for (k=0;k<N/r;k++)
      C[i][j] += A[i][k]*B[k][j]
```

- r = block (sub-matrix) size (Assume r divides N)
- $X[i][j]$ = a sub-matrix of X , defined by block row i and block column j

Great Ideas in Computer Architecture

1. *Design for Moore's Law*
 - Higher capacities caches and DRAM
2. Abstraction to Simplify Design
3. Make the Common Case Fast
4. *Dependability via Redundancy*
 - Parity, SEC/DEC
5. *Memory Hierarchy*
 - Caches, TLBs
6. *Performance via Parallelism/Pipelining/Prediction*
 - Data-level Parallelism

Course Summary

- As the field changes, Computer Architecture courses change, too!
- It is still about the software-hardware interface
 - Programming for performance!
 - Parallelism: Task-, Thread-, Instruction-, and Data-MapReduce, OpenMP, C, SSE Intrinsics
 - Understanding the memory hierarchy and its impact on application performance