ENGRI 1210
Recent Trends and Applications in Computer Engineering

Christopher Batten

School of Electrical and Computer Engineering
Cornell University
The Computer Systems Stack

Gap too large to bridge in one step
(but there are exceptions, e.g., a magnetic compass)
In its broadest definition, computer engineering is the development of the abstraction/implementation layers that allow us to execute information processing applications efficiently using available manufacturing technologies.
Cornell Computer Engineering Curriculum

- Application
- Algorithm
- Programming Language
- Operating System
- Instruction Set Architecture
- Microarchitecture
- Register-Transfer Level
- Gate Level
- Circuits
- Devices
- Technology

ECE 2400 Computer Systems Programming
ECE 3140 Embedded Systems
ECE 4760 Design with Microcontrollers
ECE 4750 Computer Architecture
ECE 2300 Digital Logic & Computer Org
Cornell Computer Engineering Curriculum

- Freshman
  - ECE 2100
  - ECE 1210

- Freshman/Sophomore
  - ECE 3140
  - CS 3320
  - ECE 3420
  - CS 2210

- Sophomore/Junior
  - ECE 3150
  - ECE 3400

- Junior/Senior M.Eng. Ph.D.
  - ECE 4740
  - ECE 4760
  - ECE 4750

- Senior M.Eng.
  - ECE 5745
  - ECE 5760
  - ECE 5770
  - ECE 5775

- Senior M.Eng. Ph.D.
  - ECE 5730
  - ECE 5750
  - ECE 5720

- Systems track
  - CS 5220

- VLSI track
  - CS 5420

- Architecture track
  - Cs 4410

- Core courses
  - Prerequisite
  - Pre- or corequisite

ENGRI 1210 Recent Trends and Applications in Computer Engineering

4 / 63
Processors, Memories, and Networks

Computer engineering basic building blocks

- **Processors** for computation
- **Memories** for storage
- **Networks** for communication
The Computer Systems Stack

Activity 1

Trends in Computer Engineering

Activity 2

Hardware Acceleration for Deep Learning

Agenda

The Computer Systems Stack

Activity 1

Trends in Computer Engineering

Activity 2

Hardware Acceleration for Deep Learning
Activity #1: Sorting with a Sequential Processor

**Application:** Sort 32 numbers

**Simulated Sequential Computing System**
- **Processor:** You!
- **Memory:** Worksheet, read input data, write output data
- **Network:** Passing/collecting the worksheets

**Activity Steps**
- 1. Discuss strategy with neighbors
- 2. When instructor starts timer, flip over worksheet
- 3. Sort 32 numbers as fast as possible
- 4. Lookup when completed and write time on worksheet
- 5. Raise hand
- 6. When everyone is finished, then analyze data
## Agenda

The Computer Systems Stack

**Activity 1**

**Activity 2**

Hardware Acceleration for Deep Learning

---

<table>
<thead>
<tr>
<th>Application</th>
<th>Algorithm</th>
<th>PL</th>
<th>OS</th>
<th>ISA</th>
<th>μArch</th>
<th>RTL</th>
<th>Gates</th>
<th>Circuits</th>
<th>Devices</th>
<th>Technology</th>
</tr>
</thead>
</table>

---

**Trends in Computer Engineering**

**Hardware Acceleration for Deep Learning**
Trend 1: Bell’s Law

Roughly every decade a new, smaller, lower priced computer class forms based on a new programming platform resulting in entire new industries.
Trend 1: Growing Diversity in Apps & Systems

- Game Consoles
- Computing: From Handhelds to Servers
- Automobiles
- Digital Cameras
- Internet Routers
- GPS Devices and Satellites
- Humanoid Robots
- Unmanned Vehicles
- Wearable Activity Monitors
- Smart Home
- Wearable Activity Monitors
- Wearable Activity Monitors
- Medical Devices
- Sensor Networks
Example: Internet of Things

$1.7 trillion
Market for IoT by 2020
— IDC

25 billion
Connected “things” by 2020
— Gartner
IoT Platform Startups

Particle: Photon
WiFi connected µcontrollers w/ Particle Cloud

Punch Through

Devices ➔ Particle Cloud ➔ Applications
IoT Chip Startups

Chip startup founded in 2014 to use ultra-low-power circuits in energy harvesting IoT devices

B. Calhoun, D. Wentzloff, et al.
Univ. of Virginia, Univ. of Michigan
IoT for Truly Personalized Medicine

- Smart Pill Boxes
- Heartbeat Sensor
- Medical Services Gateway
- Wearable Devices
- PHR
  - Patient (owner)
  - Authorized third party
- Collaborative Patient Devices
- Self-management/Monitoring
- Health Insurance, Government
- Health Providers
  - Exams and Images Laboratories
- Health organizations and professionals

- Blood Pressure
- Blood Sugar
- Weight Scales
- Mobile Devices
- Sensors/IoT
The Computer Systems Stack

Application
Algorithm
Programming Language
Operating System
Instruction Set Architecture
Microarchitecture
Register-Transfer Level
Gate Level
Circuits
Devices
Technology

Trend #2: Software/Architecture Interface Changing Radically
Trends in High-Performance Processors

![Graph showing trends in transistors, MIPS, SPECint performance, frequency, and power over time from 1975 to 2015.]

- **Transistors (Thousands)**: shows an exponential increase.
- **MIPS**: indicates a similar trend.
- **Intel P4**: depicted with a red line and dot markers.
- **DEC Alpha 21264**: shown with green markers.
- **MIPS R2K**: represented with blue markers.
- **Typical Power (W)**: orange line with triangular markers.
- **Frequency (MHz)**: teal line with square markers.
- **SPECint Performance**: blue line with diamond markers.

---

~9%/year to ~15%/year increase over time.
Transition to Multicore Processors

- **Intel Pentium 4**: Single monolithic processor
- **Cray XT3 Supercomputer**: 1024 single-core processors
- **AMD Quad-Core Opteron**: Four cores on the same die
- **IBM Blue Gene Q Supercomputer**: Thousands of 18-core processors
The Multicore “Hail Mary Pass”
Inevitable End of Moore’s Law as We Know It

Slowing Technology Scaling Means Golden Age of Design

Adapted from D. Brooks Keynote at NSF XPS Workshop, May 2015.

CMOS scaling is running out. [Colwell 2012]
Slowing Technology Scaling Means Golden Age of Design

Adapted from D. Brooks Keynote at NSF XPS Workshop, May 2015.

[Colwell 2012]
The Specialization “Hail Mary Pass”

- Trends in Computer Engineering
- Activity 2 Hardware Xcel for Deep Learning

Graph showing trends in computer engineering with data points for DEC Alpha 21264, MIPS R2K, Intel P4, AMD 4-Core Opteron, and Intel 48-Core Prototype. The graph includes axes for Transistor (Thousands), Frequency (MHz), SPECint Performance, Typical Power (W), and Number of Cores. The x-axis ranges from 1975 to 2015.

- Typical Power (W): ~9%/year
- Frequency (MHz): ~15%/year
- SPECint Performance: ~9%/year
- Number of Cores: Parallelism?
- Specialization?
Heterogeneous Systems-on-Chip

Adapted from D. Brooks Keynote at NSF XPS Workshop, May 2015.
Heterogeneous Systems-on-Chip

Out-of-Core Accelerators

Maltiel Consulting estimates

[www.anandtech.com/show/8562/chipworks-a8]

Adapted from D. Brooks Keynote at NSF XPS Workshop, May 2015.
Microsoft Catapult: FPGAs in the Data Center

- Custom FPGA board for accelerating Bing search and other workloads
- Accelerators developed with/by app developers
- Tightly integrated into Microsoft data center’s and cloud computing platforms, access gradually being given to outside developers
The Computer Systems Stack

- Application
- Algorithm
- Programming Language
- Operating System
- Instruction Set Architecture
- Microarchitecture
- Register-Transfer Level
- Gate Level
- Circuits
- Devices
- Technology

Trend #3: Technology/Architecture Interface Changing Radically
Trend 5: Emerging Device Technologies

- Vertical MOSFETs
- Graphene
- Carbon Nanotubes
- Nanorelays
- Quantum Computing
- Molecular Computing
- Memristors
- Phase-Change Mem
- Spintronics
- 3D Integration
- Nanophotonics

Examples of Emerging Technologies

Intel 3D Crosspoint Memory
Resistive memory enables very high density, non-volatile storage with fast access times

D-Wave
Quantum annealing computer suitable for solving complex optimization problems
A Carbon Nanotube “Computer”

Three Key Trends in Computer Engineering

Trend #1: Growing Diversity in Applications and Systems

Trend #2: Software/Arch Interface Changing Radically

Trend #3: Technology/Arch Interface Changing Radically

Students entering the field of computer engineering have a unique opportunity to shape the future of computing and how it will impact society.
Agenda

The Computer Systems Stack

Activity 1

Trends in Computer Engineering

Activity 2

Hardware Acceleration for Deep Learning
Activity #2: Sorting with a Parallel Processor

- **Application:** Sort 32 numbers

- **Simulated Parallel Computing System**
  - Processor: Group of 2–8 students
  - Memory: Worksheet, scratch paper
  - Network: Communicating between students

- **Activity Steps**
  1. Discuss strategy with group
  2. When instructor starts timer, master processor flips over worksheet
  3. Sort 32 numbers as fast as possible
  4. Lookup when completed and write time on worksheet
  5. *Master processor only* raises hand
  6. When everyone is finished, then analyze data
Activity #2: Discussion

Distribute

Sort 4 Numbers

Merge Phase 1
> merge 4+4 = 8

Merge Phase 2
> merge 8+8 = 16

Merge Phase 3
> merge 16+16 = 32

Algorithm
Communication
Load Balancing
Fault Tolerance
Dataset Size

unsorted
sorted
Agenda

The Computer Systems Stack

Activity 1

Trends in Computer Engineering

Activity 2

Hardware Acceleration for Deep Learning
ImageNet: Object Recognition Competition
Machine Learning: Training vs. Inference
An Inflection Point due to Algorithms and Hardware

The Onyx2 differs significantly from the shared memory multiprocessor system of the original Onyx. The key design tradeoff for InfiniteReality is that the graphics system must be capable of working on two generations of host platforms. The first generation of InfiniteReality used a distributed shared memory multiprocessor system with cache-coherent non-uniform memory access. The second generation, which is based on the Onyx, uses a distributed shared memory multiprocessor system with cache-coherent non-uniform memory access.

The heart of the Geometry Engine is a single instruction multiple data (SIMD) arrangement of three floating point cores, each of which comprises an ALU and a multiplier plus a 32 word register file. The host interface processor is a 40 MHz Intel 80186 microprocessor, which communicates with the host interface board and the geometry board. The geometry board is a distributed shared memory multiprocessor system with cache-coherent non-uniform memory access. The geometry board contains four geometry engines and a geometry-Raster FIFO. The geometry-Raster FIFO is used to recreate the original order of incoming primitives.

The display generator board contains hardware to drive up to eight display output channels, each with its own video framebuffer. The display generator board is connected to the host interface processor via a host system bus. The host interface processor communicates with the geometry board via a vertex bus. The geometry board is connected to the display generator board via an image engine bus.

The display generator board contains hardware to drive up to eight display output channels, each with its own video framebuffer. The display generator board is connected to the host interface processor via a host system bus. The host interface processor communicates with the geometry board via a vertex bus. The geometry board is connected to the display generator board via an image engine bus.

NVIDIA GeForce 6800

NVIDIA G80

What is in a GPU?

A GPU is a heterogeneous chip multi-processor (highly tuned for graphics)

Adapted from K. Fatahalian, “Beyond Programmable Shading Course,” ACM SIGGRAPH 2010.
Compiling a Shader

1 unshaded fragment input record

```c
sampler mySamp;
Texture2D<float3> myTex;
float3 lightDir;

float4 diffuseShader(float3 norm, float2 uv)
{
    float3 kd;
    kd = myTex.Sample(mySamp, uv);
    kd *= clamp(dot(lightDir, norm), 0.0, 1.0);
    return float4(kd, 1.0);
}
```

1 shaded fragment output record

Adapted from K. Fatahalian, “Beyond Programmable Shading Course,” ACM SIGGRAPH 2010.
Executing a Shader

Adapted from K. Fatahalian, “Beyond Programmable Shading Course,” ACM SIGGRAPH 2010.

Execute shader:  
\[
\begin{align*}
\text{<diffuseShader>:} \\
\text{sample r0, v4, t0, s0} \\
\text{mul r3, v0, cb0[0]} \\
\text{madd r3, v1, cb0[1], r3} \\
\text{madd r3, v2, cb0[2], r3} \\
\text{clmp r3, r3, l(0.0), l(1.0)} \\
\text{mul o0, r0, r3} \\
\text{mul o1, r1, r3} \\
\text{mul o2, r2, r3} \\
\text{mov o3, l(1.0)}
\end{align*}
\]
“CPU-Style” Cores

Adapted from K. Fatahalian, “Beyond Programmable Shading Course,” ACM SIGGRAPH 2010.
Slimming Down

Idea #1:
Remove components that help a single instruction stream run fast

Adapted from K. Fatahalian, “Beyond Programmable Shading Course,” ACM SIGGRAPH 2010.
Two Cores (Execute Two Fragments in Parallel)

Adapted from K. Fatahalian, “Beyond Programmable Shading Course,” ACM SIGGRAPH 2010.
Four Cores (Execute Four Fragments in Parallel)

Adapted from K. Fatahalian, “Beyond Programmable Shading Course,” ACM SIGGRAPH 2010.
16 Cores (Execute 16 Fragments in Parallel)

16 cores = 16 simultaneous instruction streams

Adapted from K. Fatahalian, “Beyond Programmable Shading Course,” ACM SIGGRAPH 2010.
Instruction Stream Sharing

But ... many fragments should be able to share an instruction stream!

Adapted from K. Fatahalian, “Beyond Programmable Shading Course,” ACM SIGGRAPH 2010.
Recall: Simple Processing Core

Adapted from K. Fatahalian, “Beyond Programmable Shading Course,” ACM SIGGRAPH 2010.
Add ALUs

Idea #2: Amortize cost/complexity of managing an instruction stream across many ALUs

**SIMD processing**

Adapted from K. Fatahalian, “Beyond Programmable Shading Course,” ACM SIGGRAPH 2010.
Execute 128 Fragments in Parallel

Adapted from K. Fatahalian, “Beyond Programmable Shading Course,” ACM SIGGRAPH 2010.
But What About Branches?

Adapted from K. Fatahalian, “Beyond Programmable Shading Course,” ACM SIGGRAPH 2010.
But What About Memory Stalls?

Adapted from K. Fatahalian, “Beyond Programmable Shading Course,” ACM SIGGRAPH 2010.
Key is Throughput!

Increase run time of one group to increase throughput of many groups

Adapted from K. Fatahalian, “Beyond Programmable Shading Course,” ACM SIGGRAPH 2010.
18 Contexts

Adapted from K. Fatahalian, “Beyond Programmable Shading Course,” ACM SIGGRAPH 2010.
Complete GPU

16 cores

8 mul-add ALUs per core
(128 total)

16 simultaneous
instruction streams

64 concurrent (but interleaved)
instruction streams

512 concurrent fragments

= 256 GFLOPs (@ 1GHz)

Adapted from K. Fatahalian, “Beyond Programmable Shading Course,” ACM SIGGRAPH 2010.
Complete "Big" GPU

32 cores, 16 ALUs per core (512 total) = 1 TFLOP (@ 1 GHz)

Adapted from K. Fatahalian, “Beyond Programmable Shading Course,” ACM SIGGRAPH 2010.
Using GPUs for General-Purpose Computing

int main( int argc, char* argv[] )
{
    // ... copy data to GPGPU ...
    vvadd<<<block_count,threads_per_block >>>
        ( dest, src0, src1, n );
    // ... copy data from GPGPU to CPU ...
}

__global__ void vvadd
    ( int dest[], int src0[], int src1[], int n )
{
    int i = blockDim.x * blockIdx.x + threadIdx.x;
    if ( i < n )
        z[i] = x[i] + y[i];
}
NVIDIA DGX-1 for Deep Learning Training
NVIDIA PX 2 for Deep Learning Inference

- Dual Next Generation Tegra
- Dual Discrete GPUs
- 12 CPU Cores
- Pascal GPU
- 8TFLOPS (FP32)
- 24DL TOPS
- 12 simultaneous LVDS camera inputs

Dual Tegras on Top
Dual Discrete GPUs on the Bottom
Liquid Cooled if All Devices used
Hardware Xcel for ML is Significant Growth Area

Hardware ML Startups

- Graphcore
- Wave Computing
- Cerebras
- Mobileye (Intel)
- Nervana (Intel)
- Movidius (Intel)

Google’s TPU

- Custom chip for accelerating Google’s TensorFlow library
- Tightly integrated into Google’s data centers
Take-Away Points

- We are entering an **exciting new era of computer engineering**
  - Growing diversity in applications & systems
  - Radical rethinking of software/architecture interface
  - Radical rethinking of technology/architecture interface

- This era offers tremendous challenges and opportunities, which makes it a **wonderful time to study and contribute to the field of computer engineering**
ECE 2400 Computer Systems Programming

- **Part 1: Basic Data Structures and Algorithms with C**
  - static typing, functions, control flow, arrays, strings, pointers, dynamic memory management
  - recursion, divide-and-conquer, dynamic programming
  - sorting, lists, stacks, queues, sets, maps

- **Part 2: Advanced Data Structures and Algorithms with C++**
  - objects, inheritance, polymorphism, templates
  - binary search trees, priority queues, hash tables, graphs, spanning trees

- **Part 3: Systems Programming with C/C++**
  - POSIX I/O, processes, threads

---

```cpp
template < typename T >
T* find_max( T* array, int n )
{
  if ( n == 0 )
    return NULL;
  T* result = &array[0];
  for ( int i = 1; i < n; i++ ) {
    if ( array[i] > *result )
      result = &array[i];
  }
}
```

Programming Assignments

- Version control
- Test-driven design
- Continuous integration
- Debugging & profiling
- Code optimization