ENGRI 1210
Recent Trends and Applications in Computer Engineering

Christopher Batten

School of Electrical and Computer Engineering
Cornell University

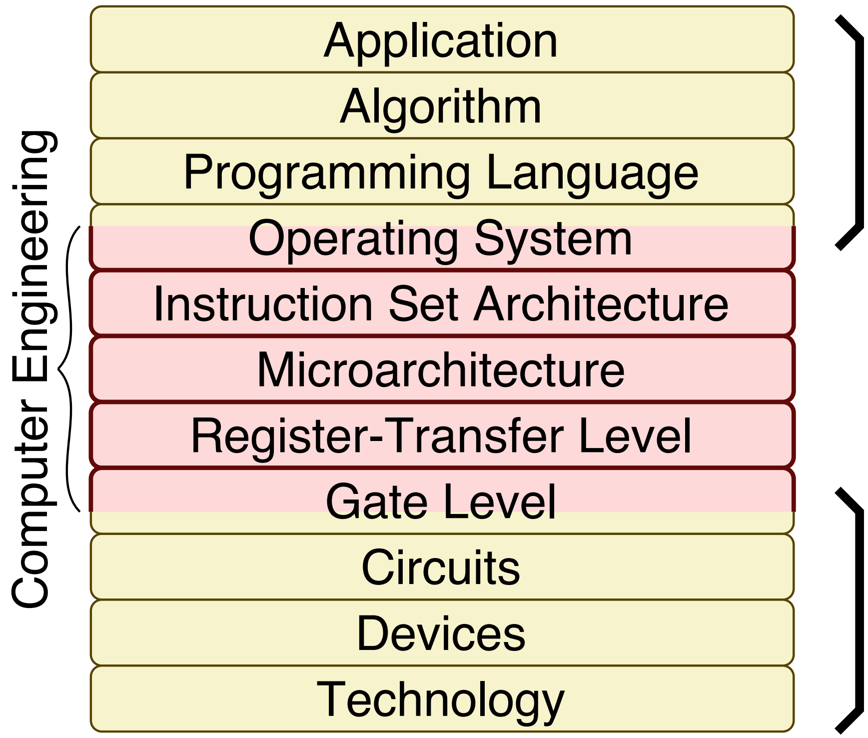
The Computer Systems Stack

Application

Technology

Gap too large to bridge in one step
(but there are exceptions,
e.g., a magnetic compass)

The Computer Systems Stack



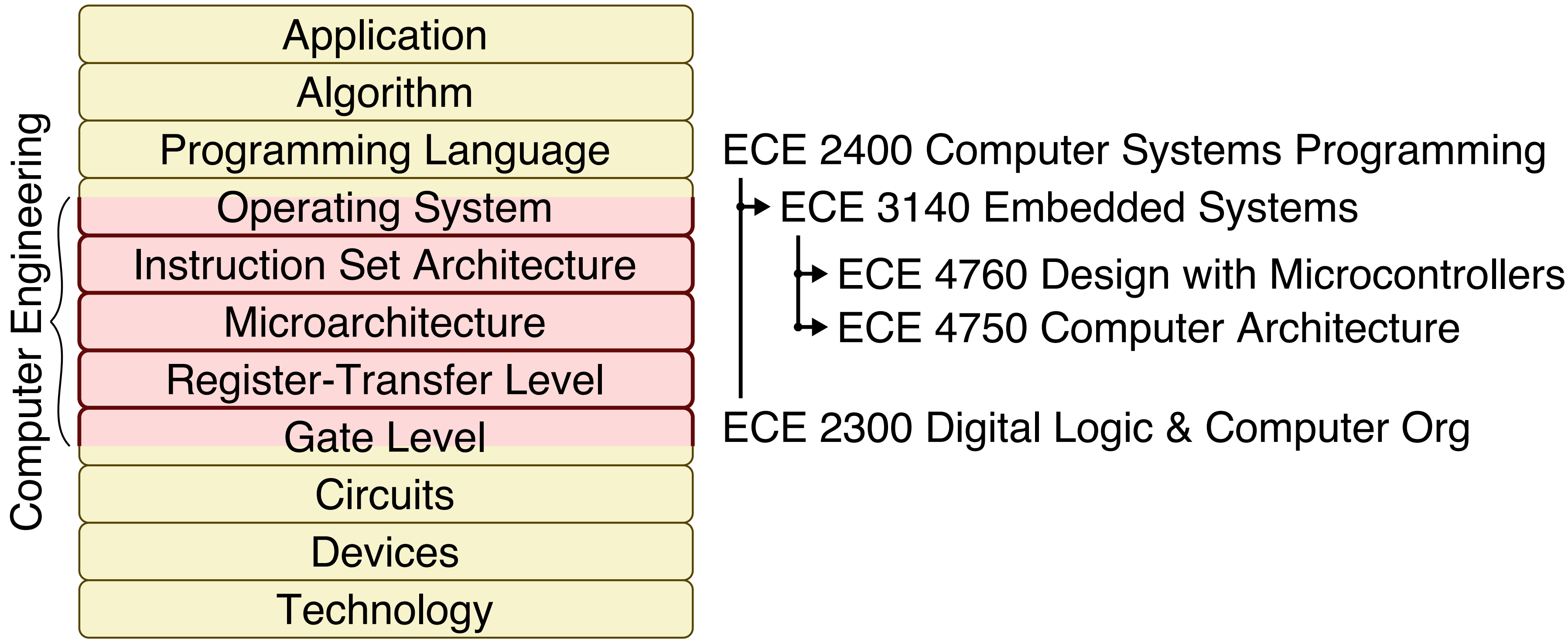
Traditional Computer Science

Computer Engineering is at the interface between hardware and software and considers the entire system

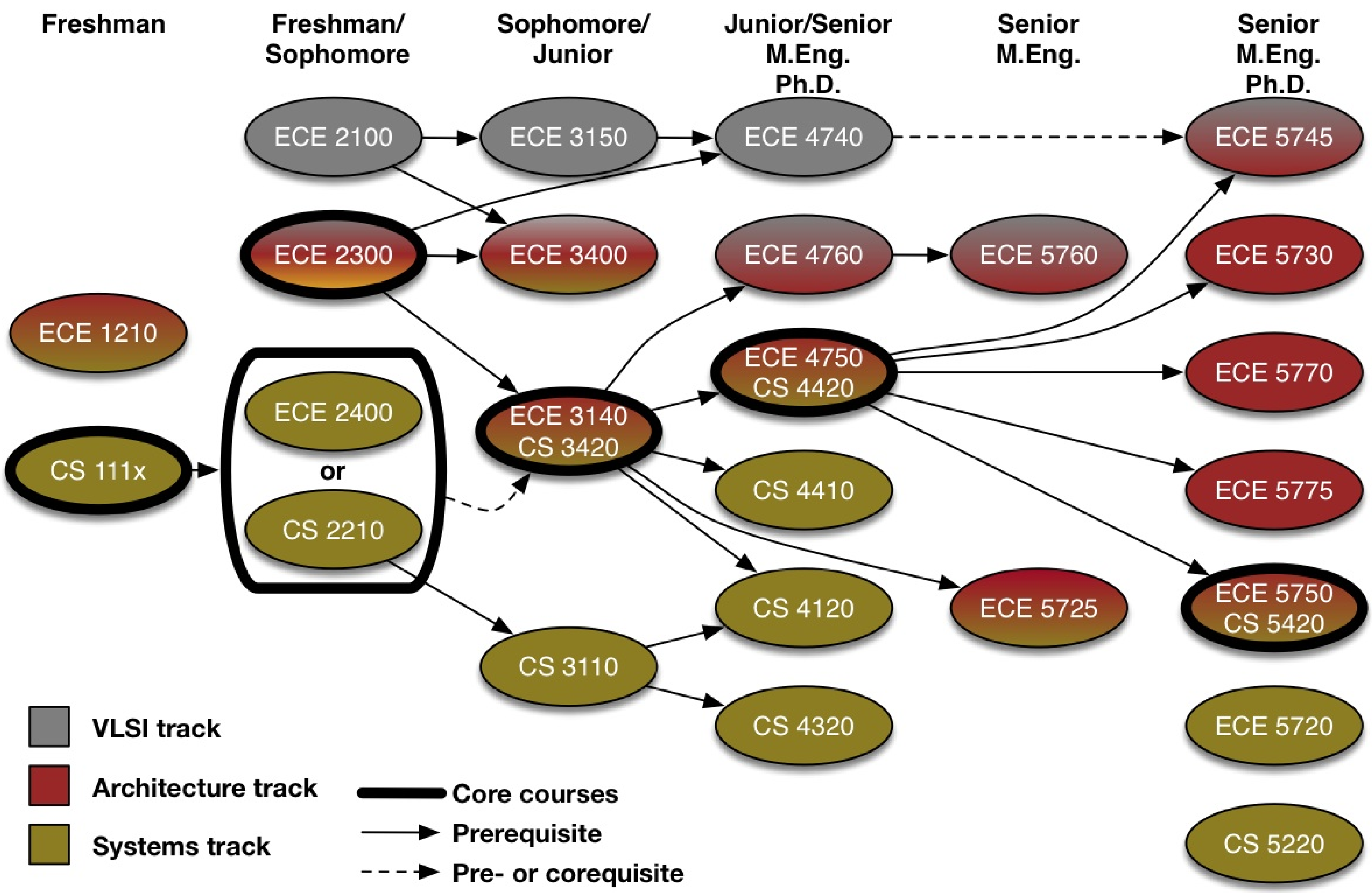
Traditional Electrical Engineering

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

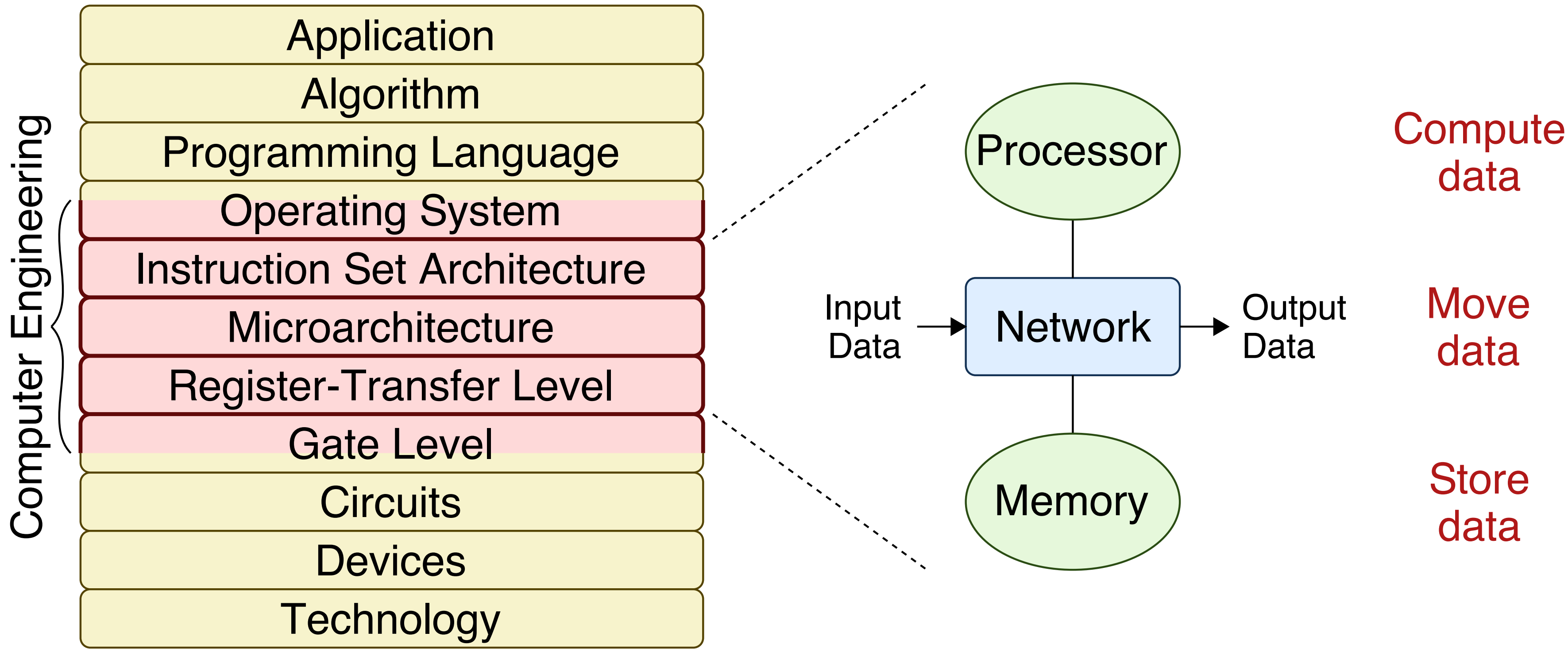
Cornell Computer Engineering Curriculum



Cornell Computer Engineering Curriculum

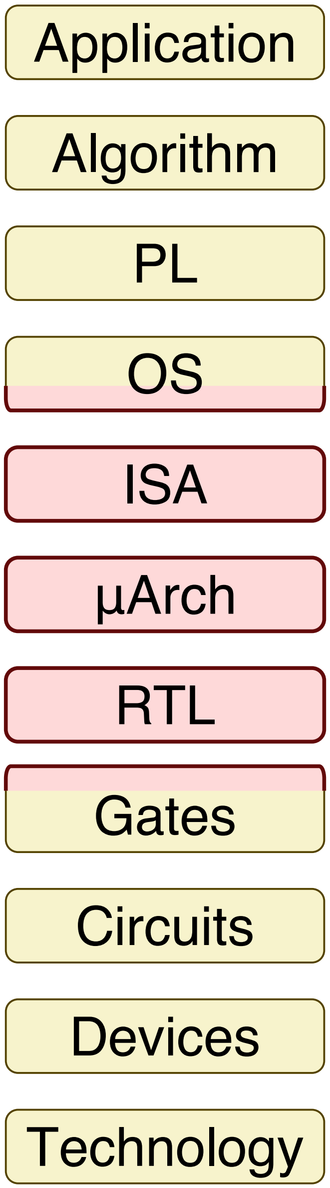


Processors, Memories, and Networks



Computer engineering basic building blocks

- Processors for computation
- Memories for storage
- Networks for communication



Agenda

The Computer Systems Stack

Activity 1

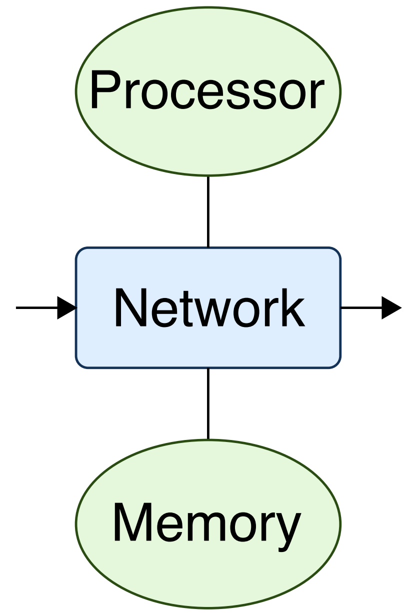
Trends in Computer Engineering

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

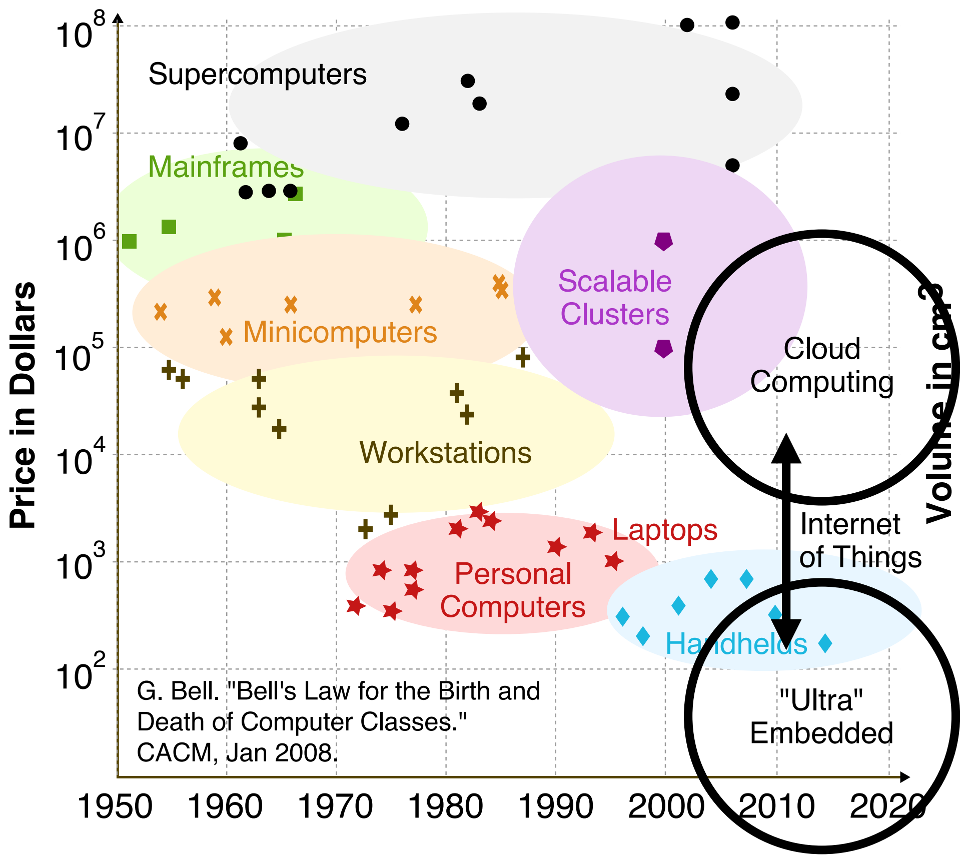


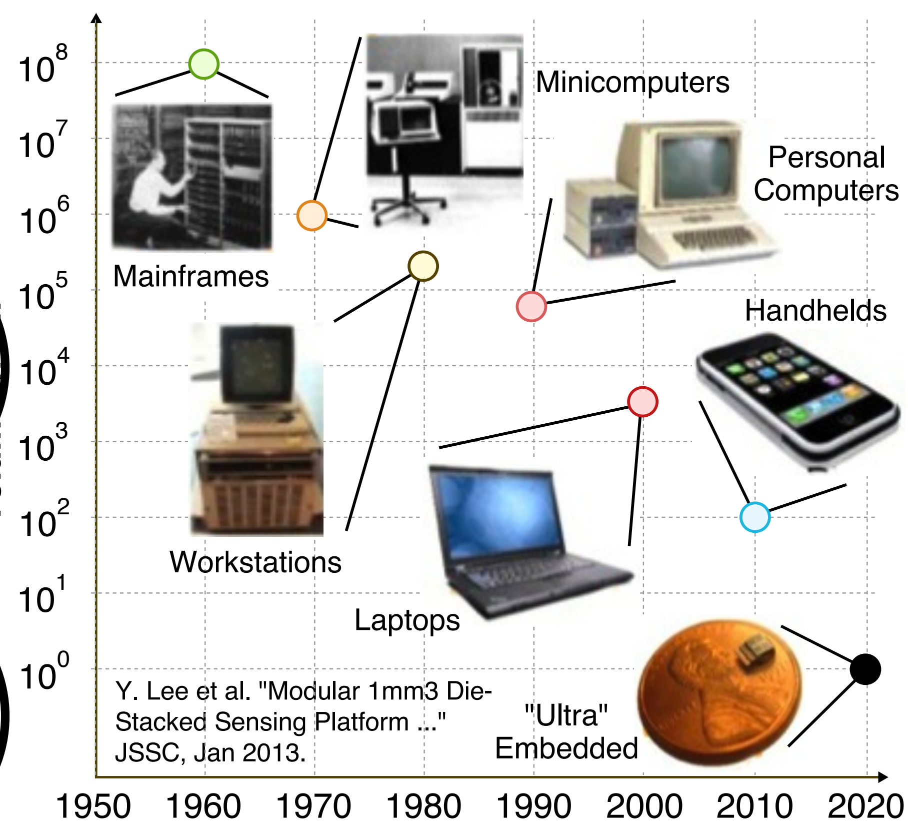
Application
Algorithm
PL
OS
ISA
µArch
RTL
Gates
Circuits
Devices
Technology

Agenda

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

Example: Internet of Things

$1.7 trillion

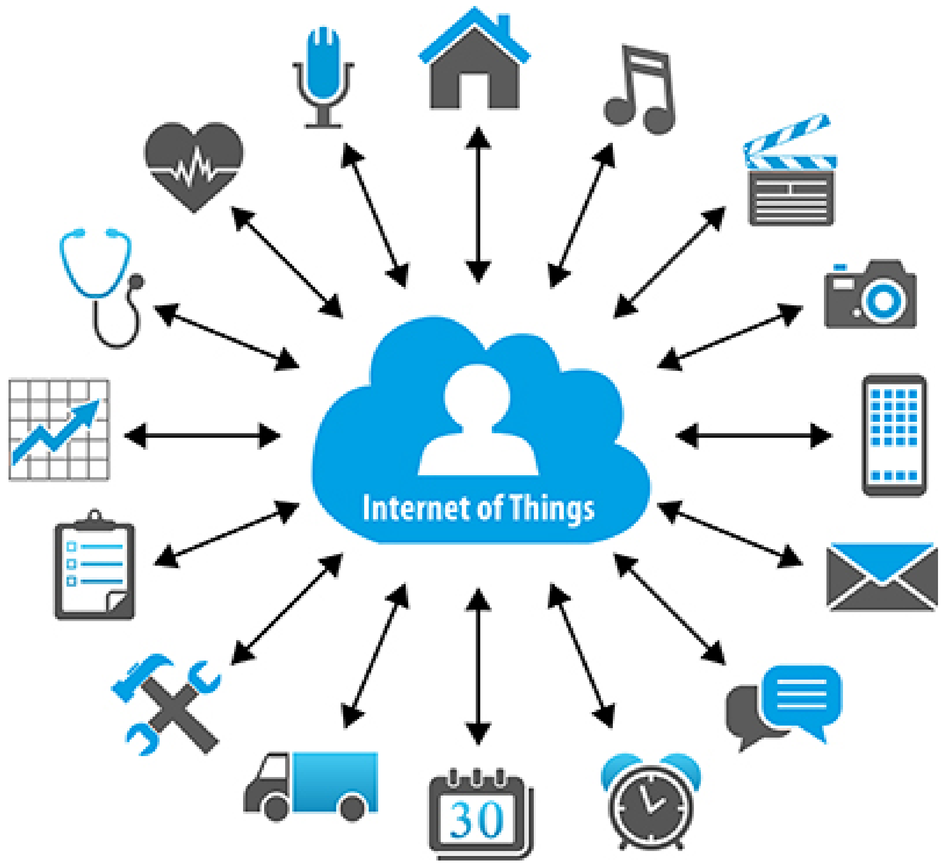
Market for IoT by 2020

— IDC

25 billion

Connected "things" by 2020

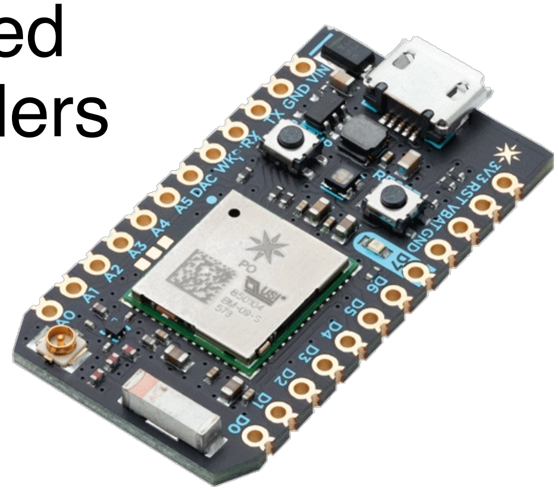
— Gartner



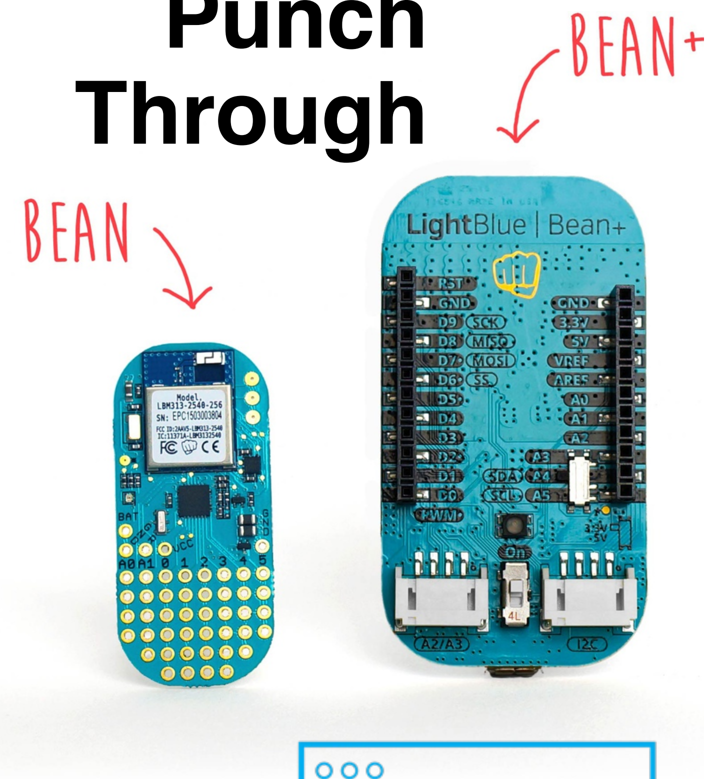
IoT Platform Startups

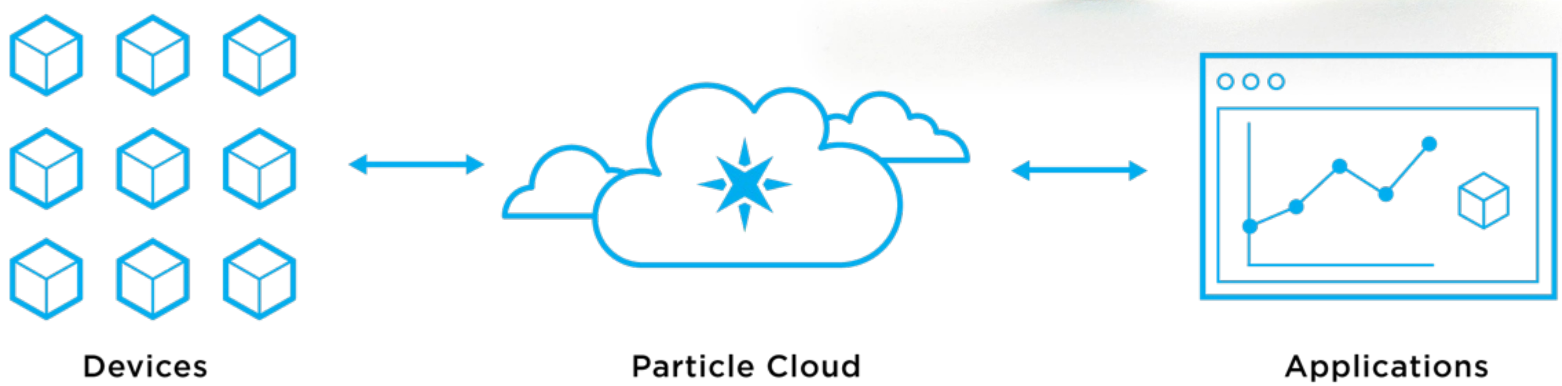
Particle: Photon

WiFi connected µcontrollers w/ Particle Cloud

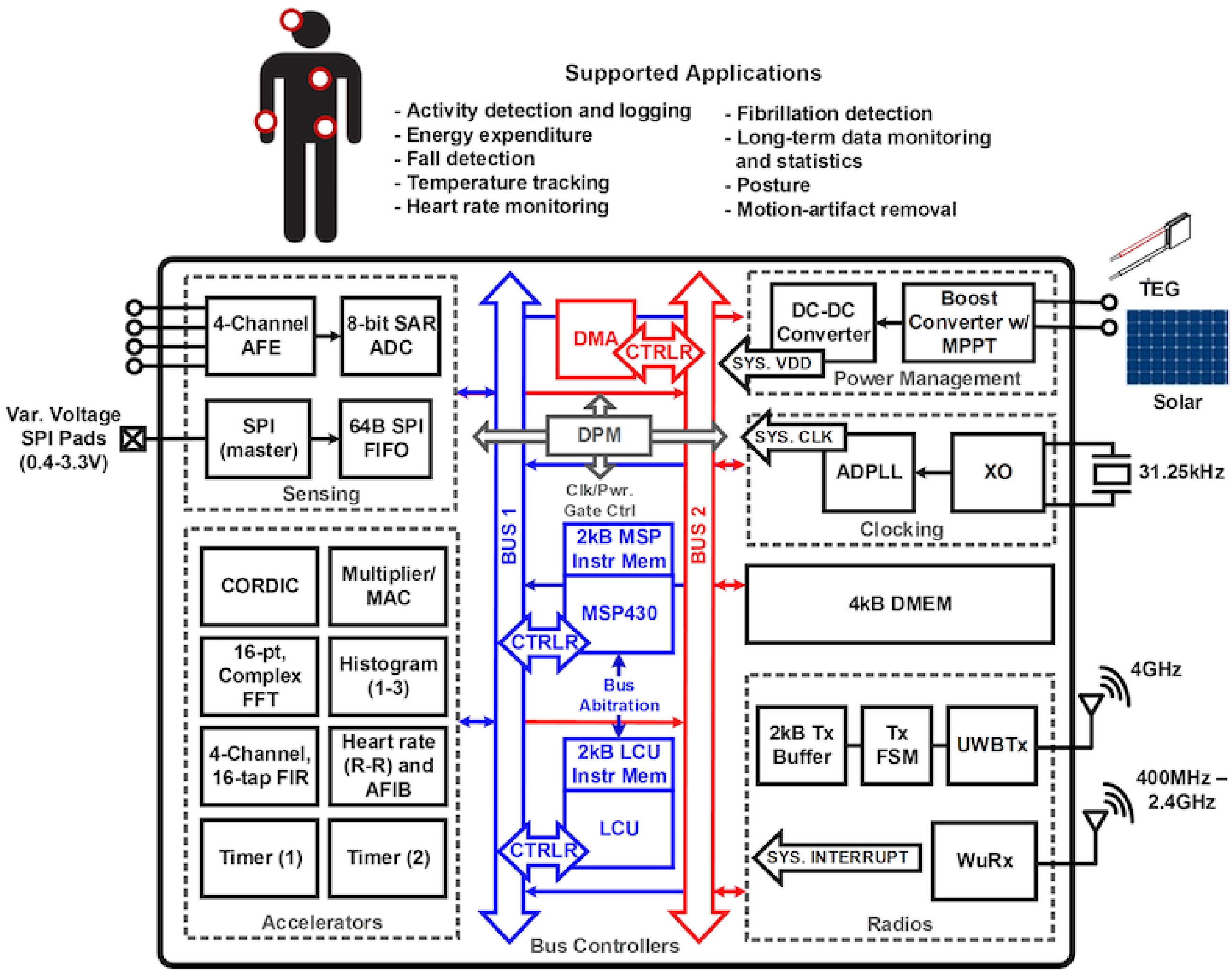


Punch Through





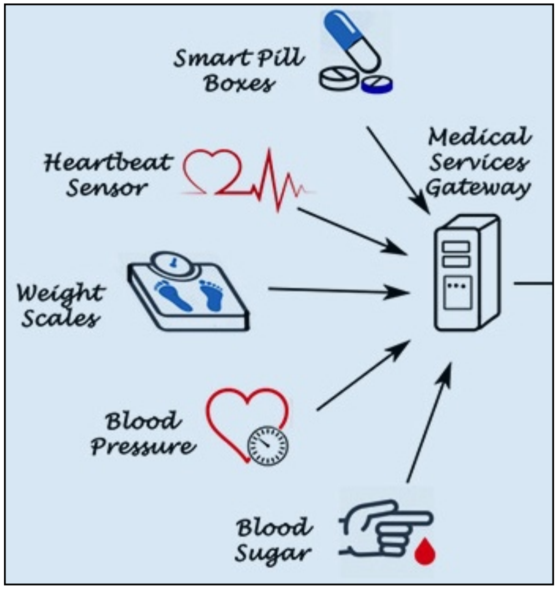
IoT Chip Startups

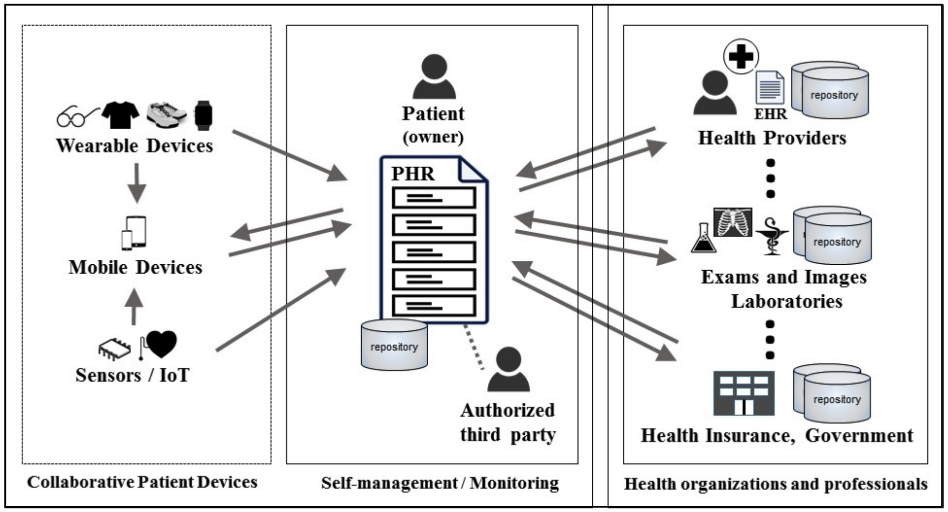


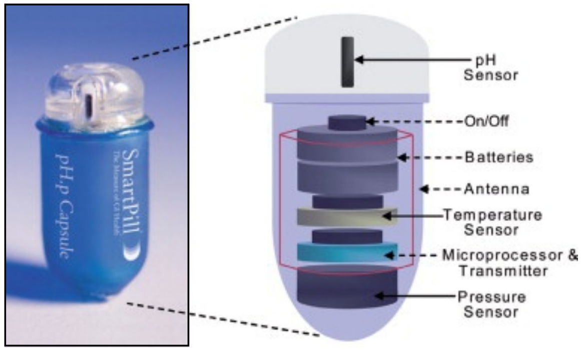
B. Calhoun, D. Wentzloff, et al.
Univ. of Virginia, Univ. of Michigan

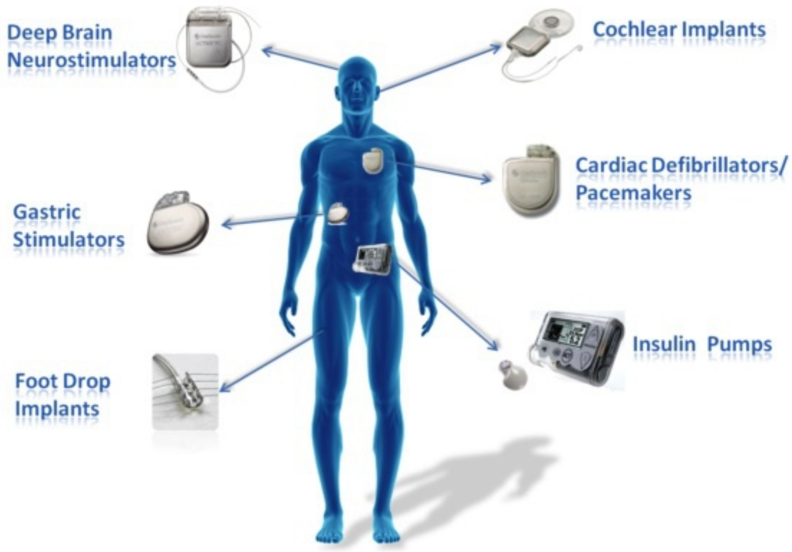
Chip startup founded in 2014 to use ultra-low-power circuits in energy harvesting IoT devices

IoT for Truly Personalized Medicine

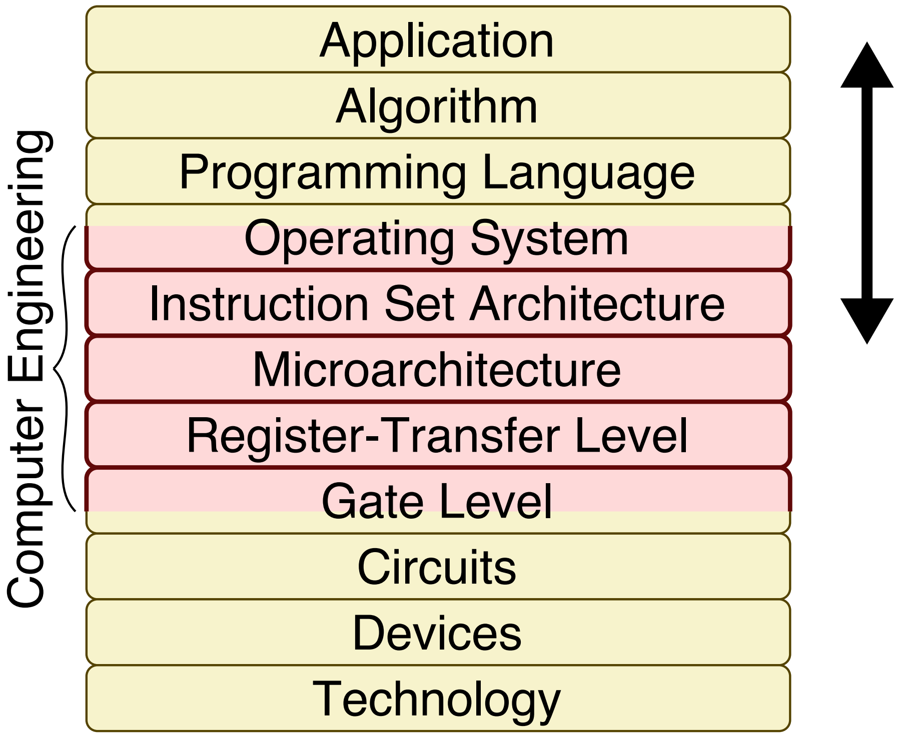






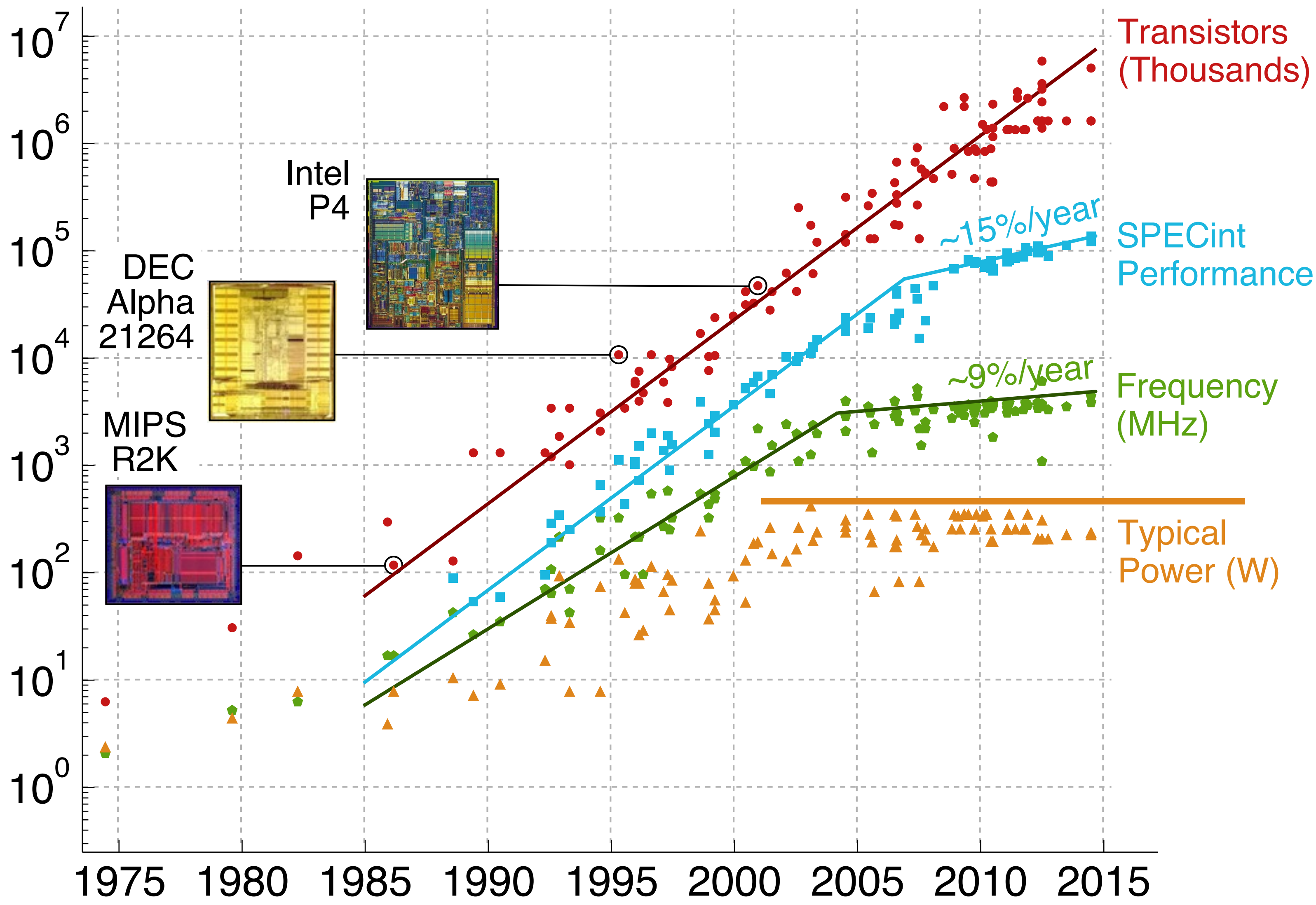


The Computer Systems Stack



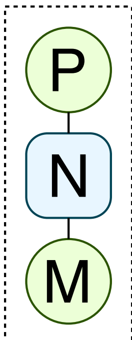
Trend #2: Software/Architecture Interface Changing Radically

Trends in High-Performance Processors

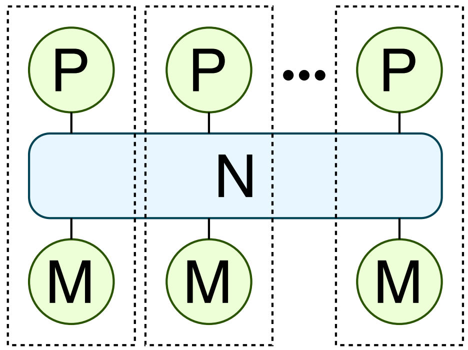


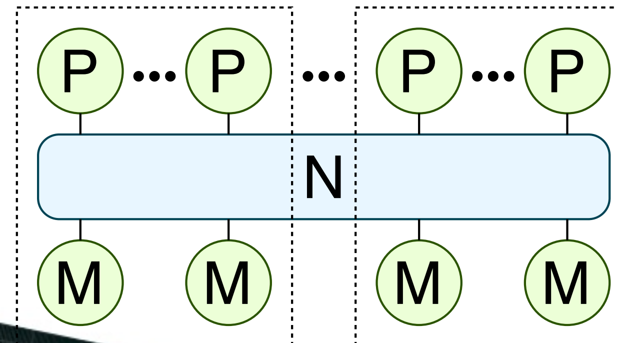
Transition to Multicore Processors

Intel Pentium 4
Single monolithic processor

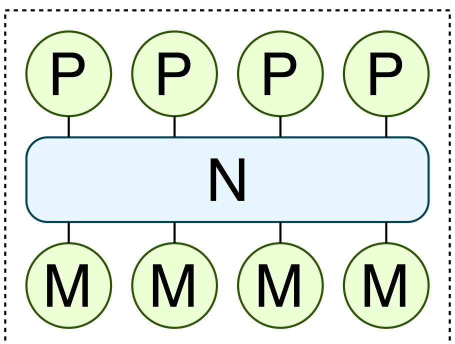


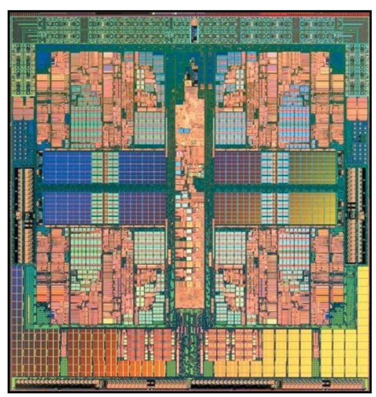
Cray XT3 Supercomputer
1024 single-core processors





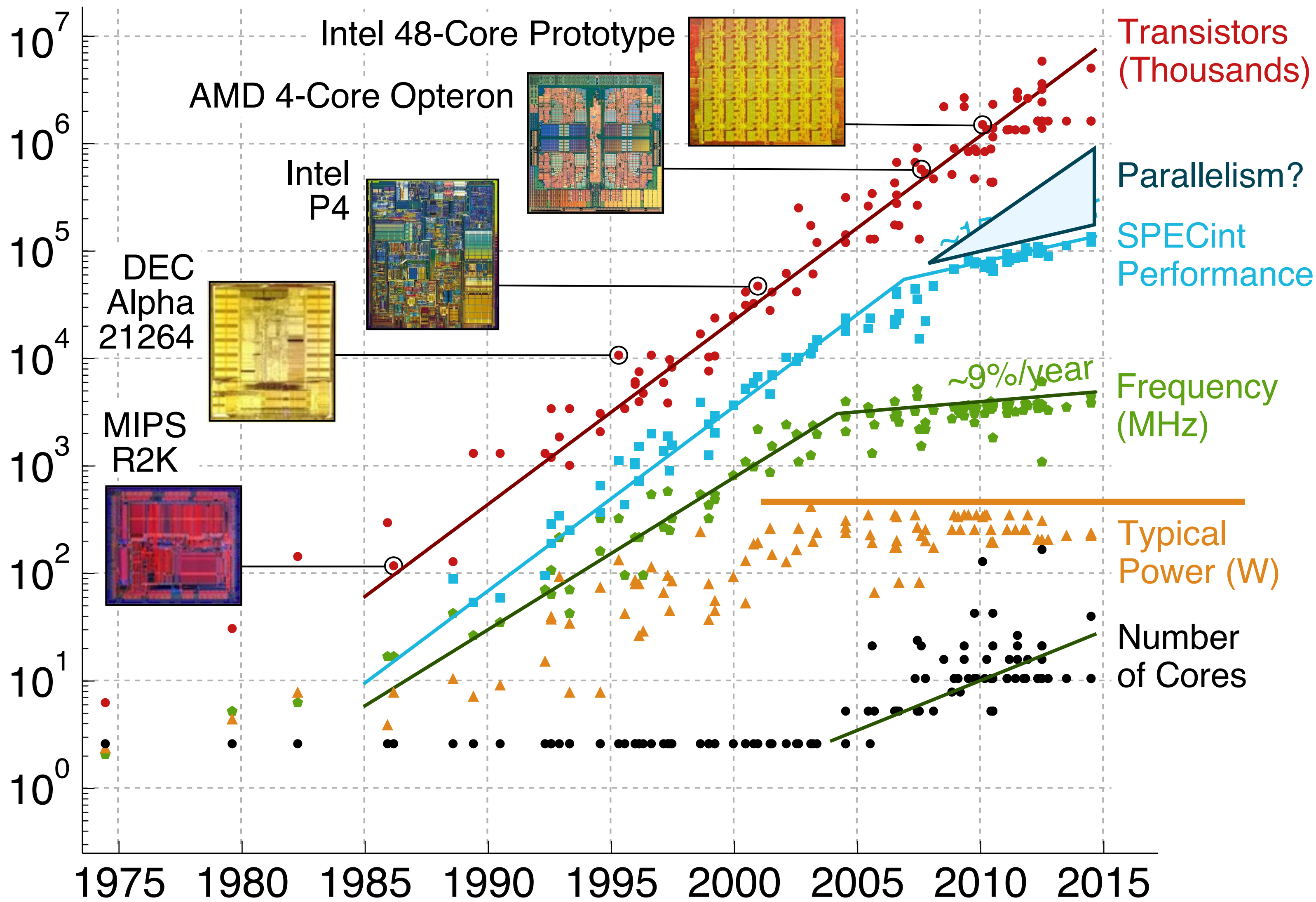
AMD Quad-Core Opteron
Four cores on the same die



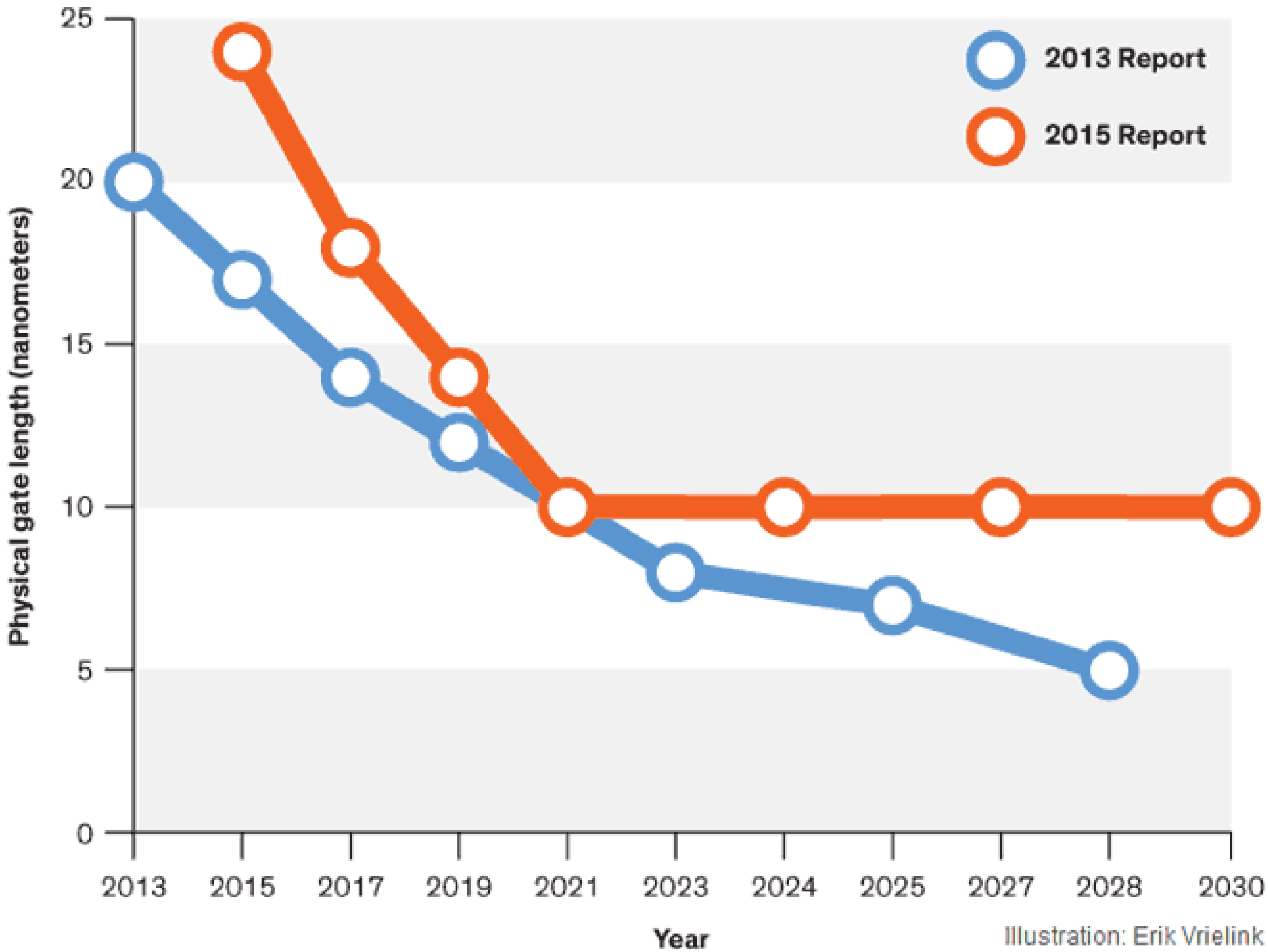


IBM Blue Gene Q Supercomputer
Thousands of 18-core processors

The Multicore “Hail Mary Pass”

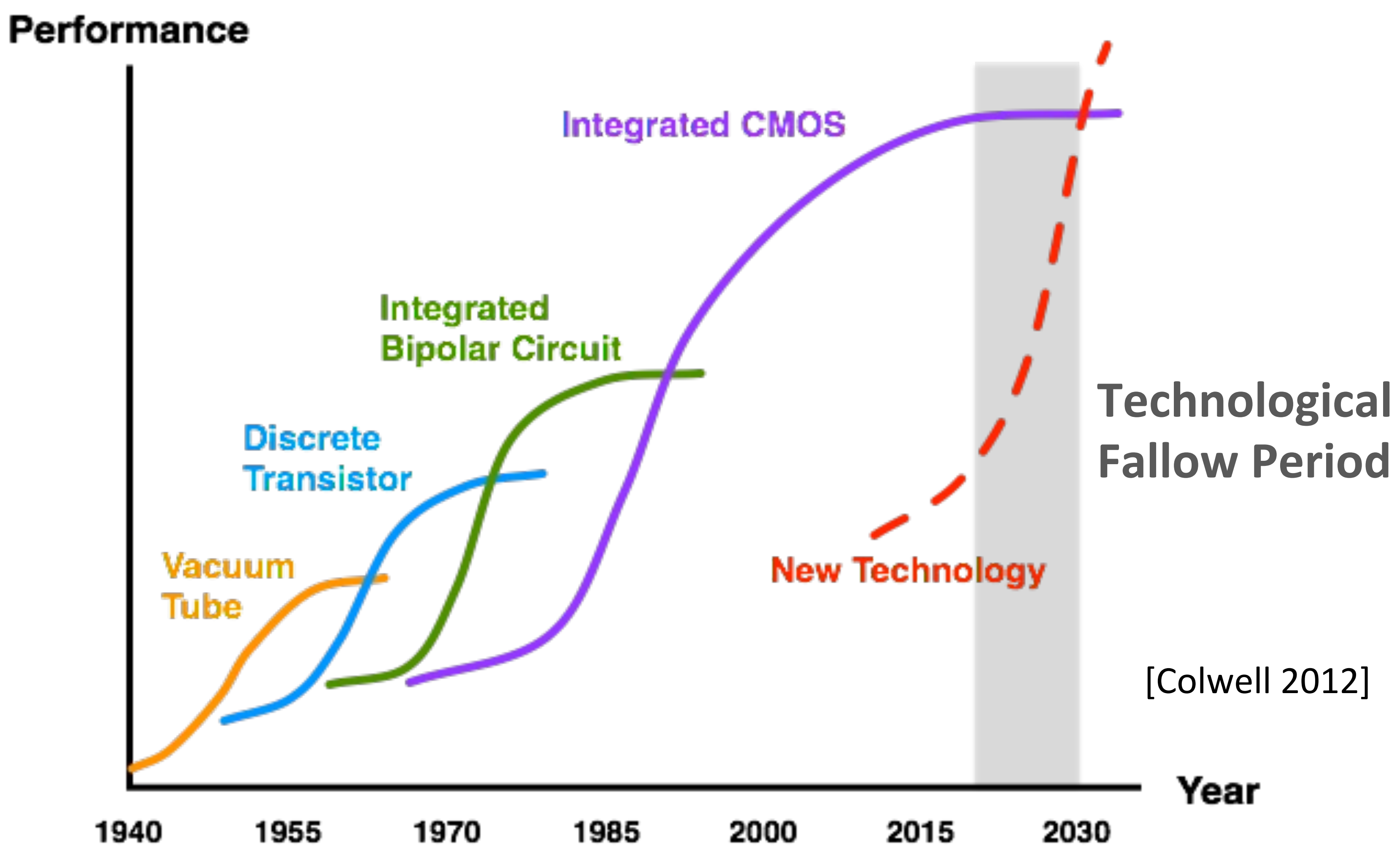


Inevitable End of Moore's Law as We Know It



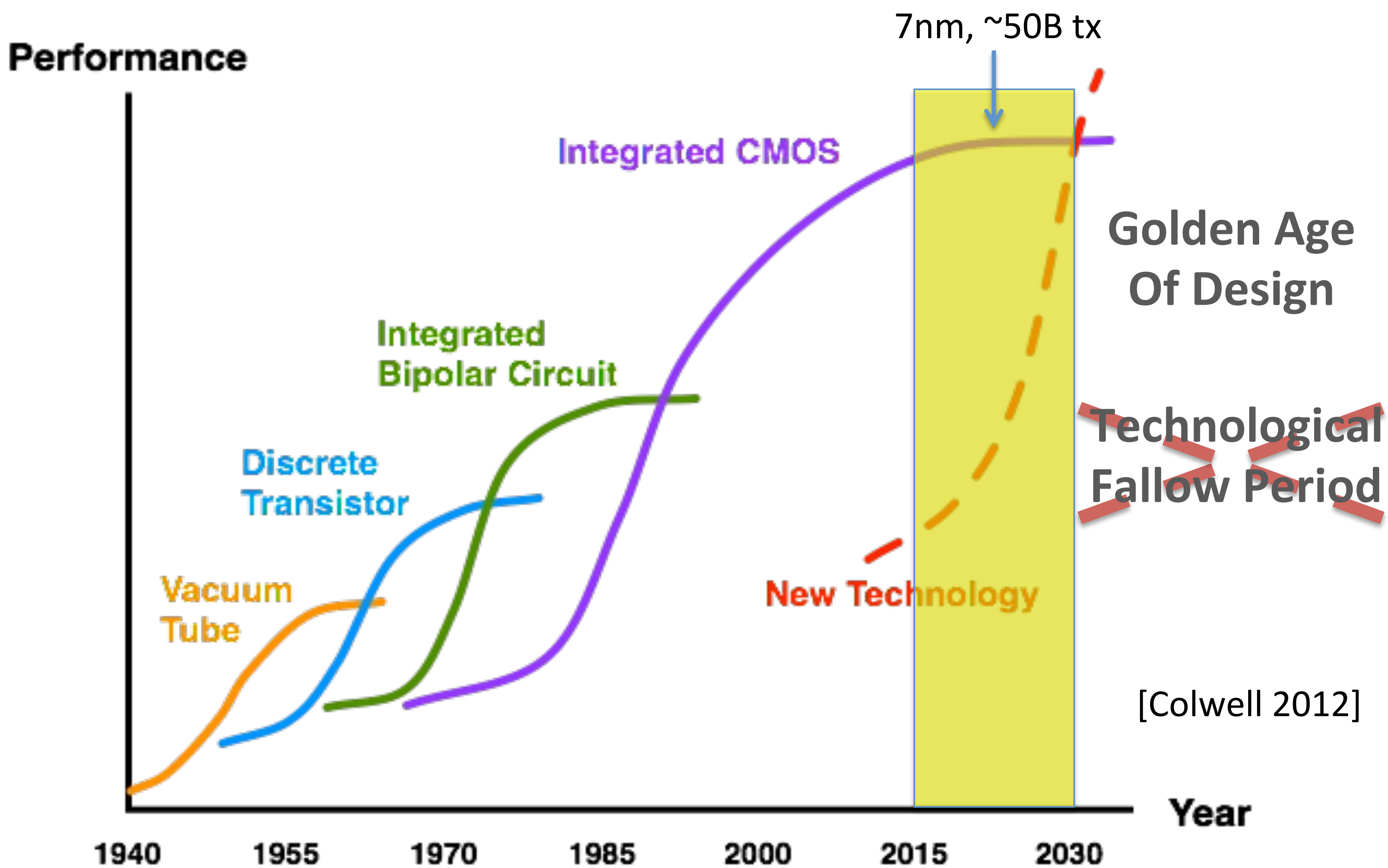
Adapted from R. Courtland, "Transistors Could Stop Shrinking in 2021," IEEE Spectrum, 2016.

Slowing Technology Scaling Means Golden Age of Design



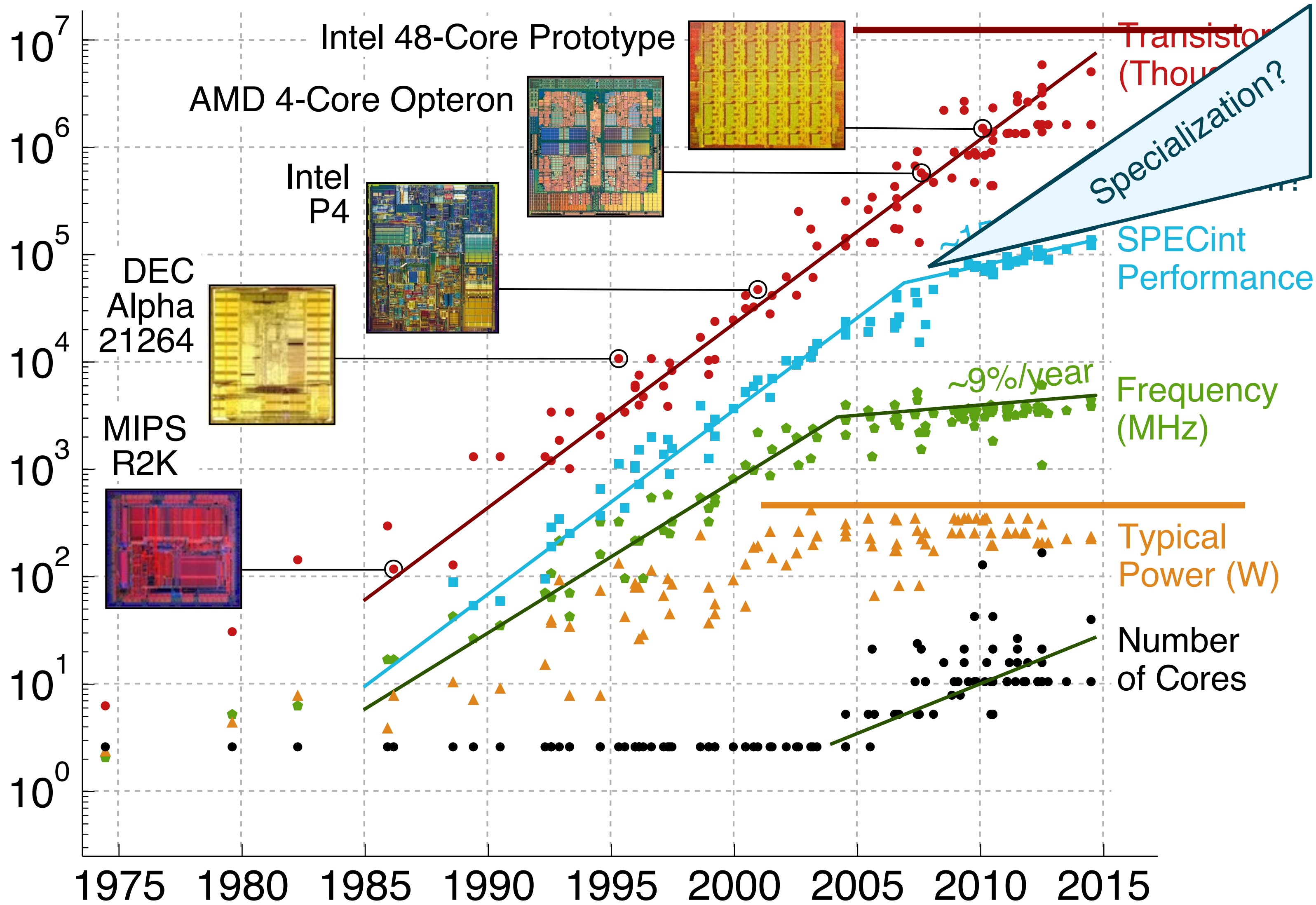
Adapted from D. Brooks Keynote at NSF XPS Workshop, May 2015.

Slowing Technology Scaling Means Golden Age of Design

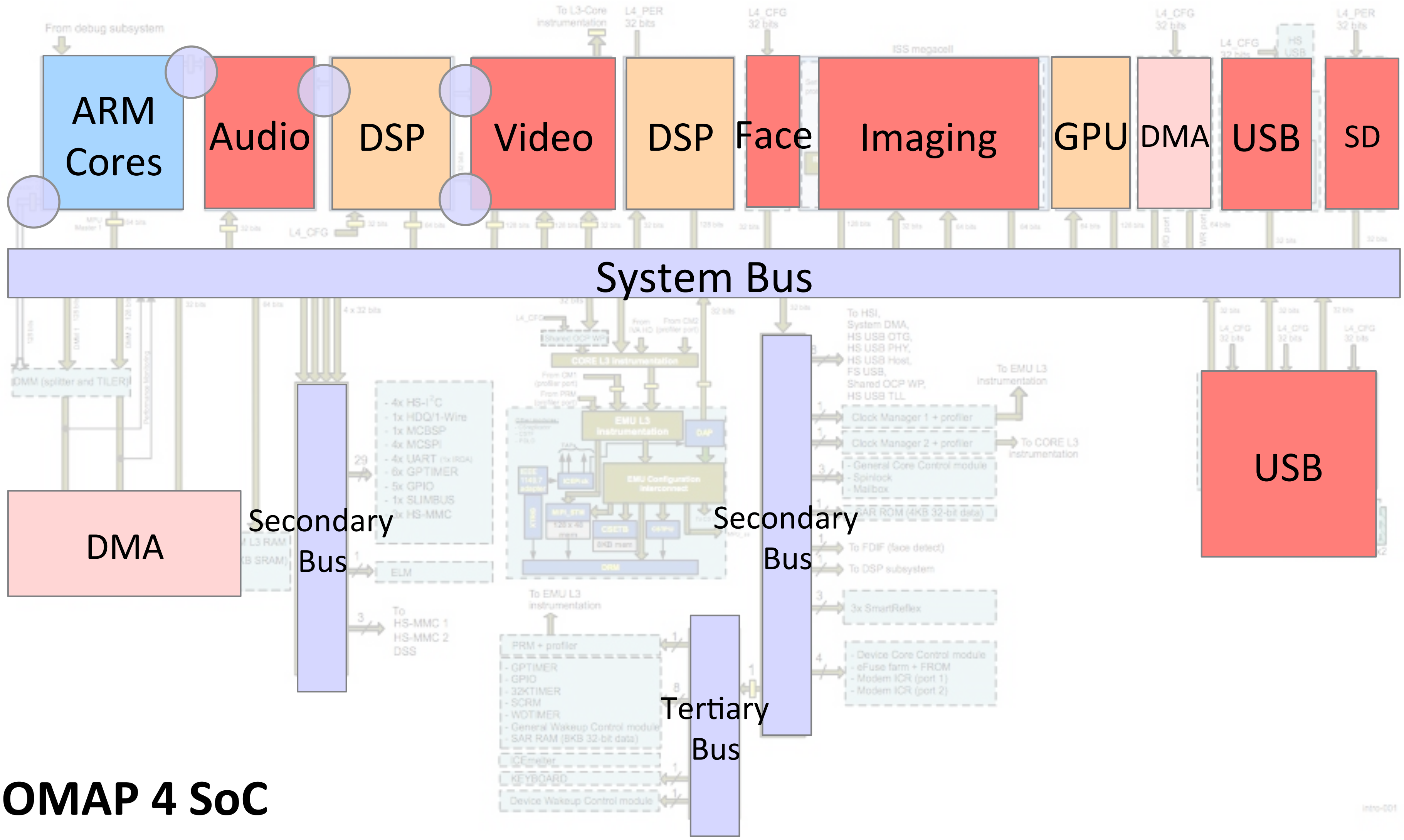


Adapted from D. Brooks Keynote at NSF XPS Workshop, May 2015.

The Specialization “Hail Mary Pass”



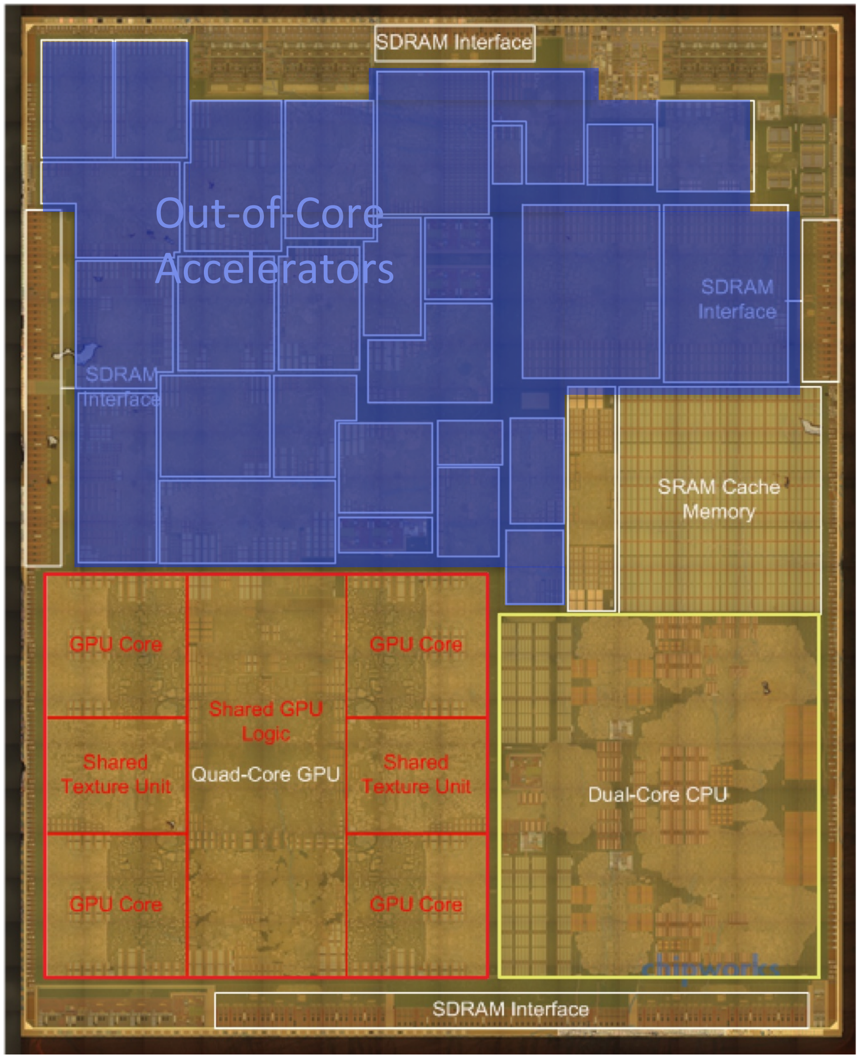
Heterogeneous Systems-on-Chip

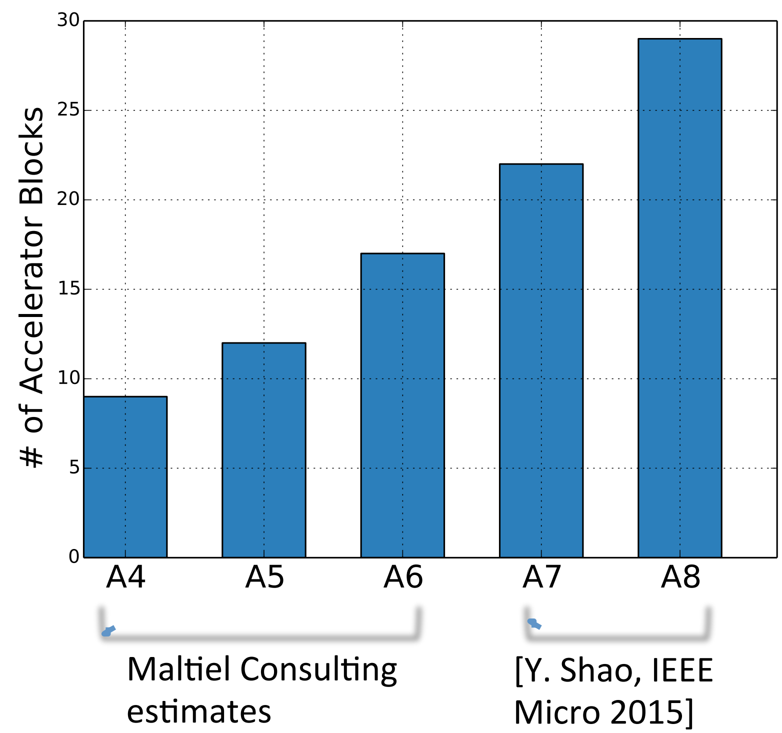


OMAP 4 SoC

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

Heterogeneous Systems-on-Chip

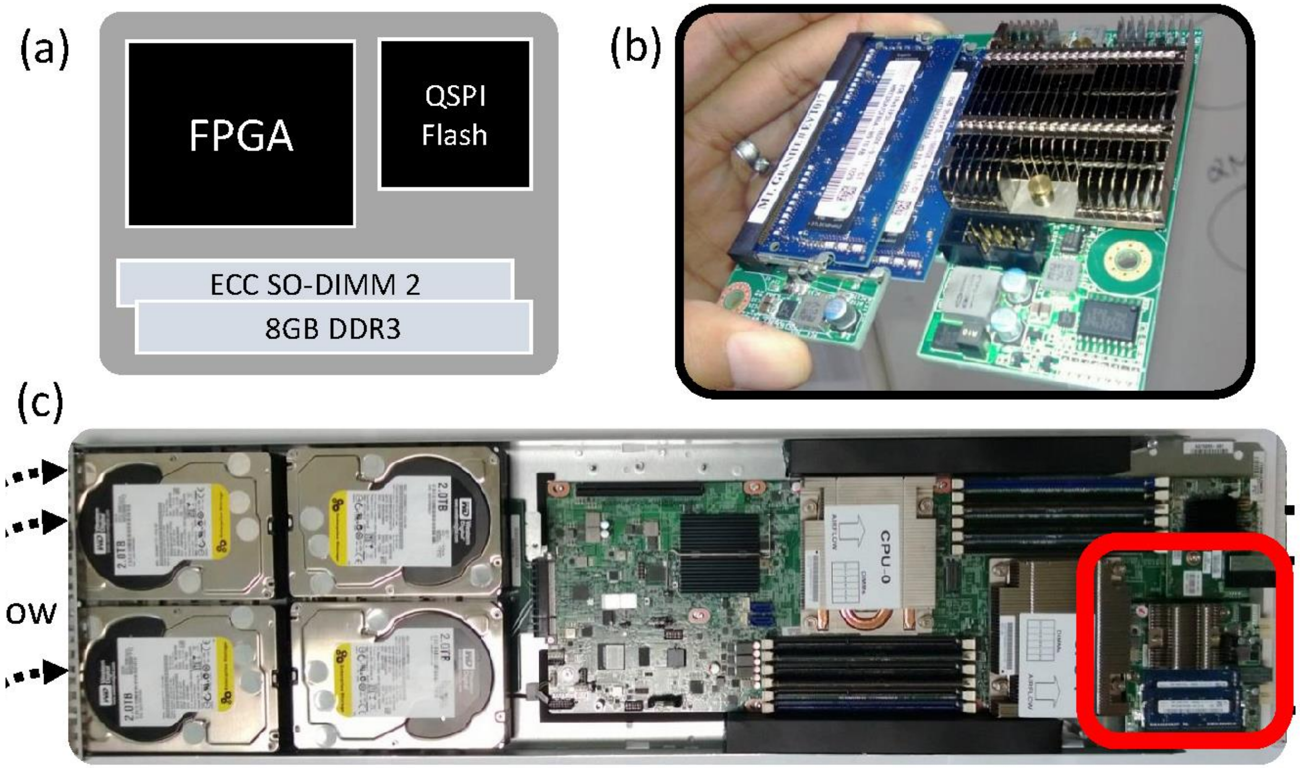




[www.anandtech.com/show/8562/chipworks-a8]

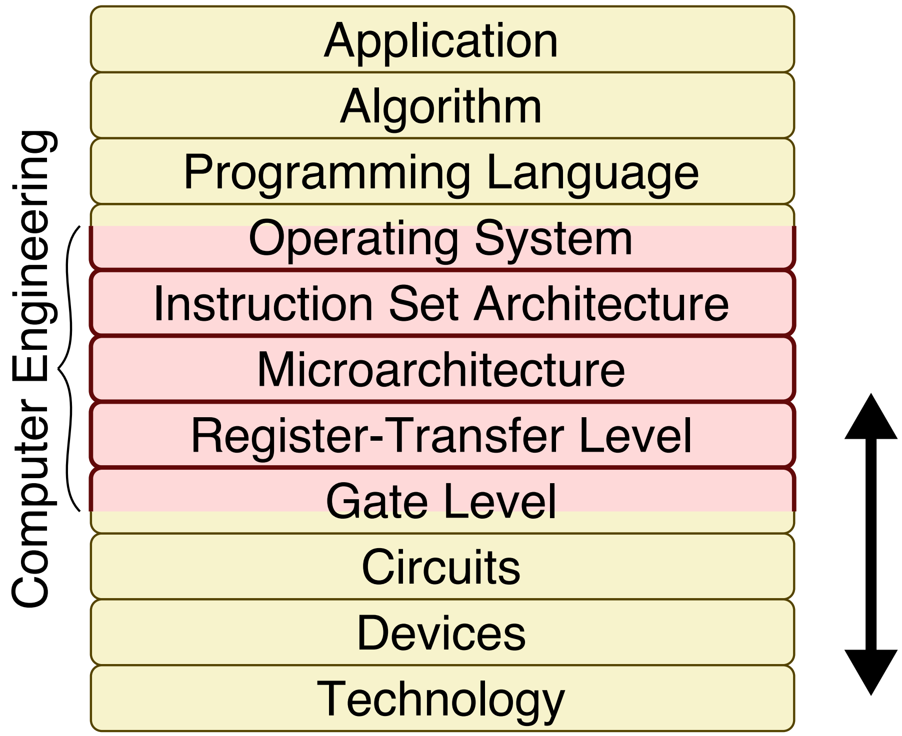
Adapted from D. Brooks Keynote at NSF XPS Workshop, May 2015.

Microsoft Catapult: FPGAs in the Data Center



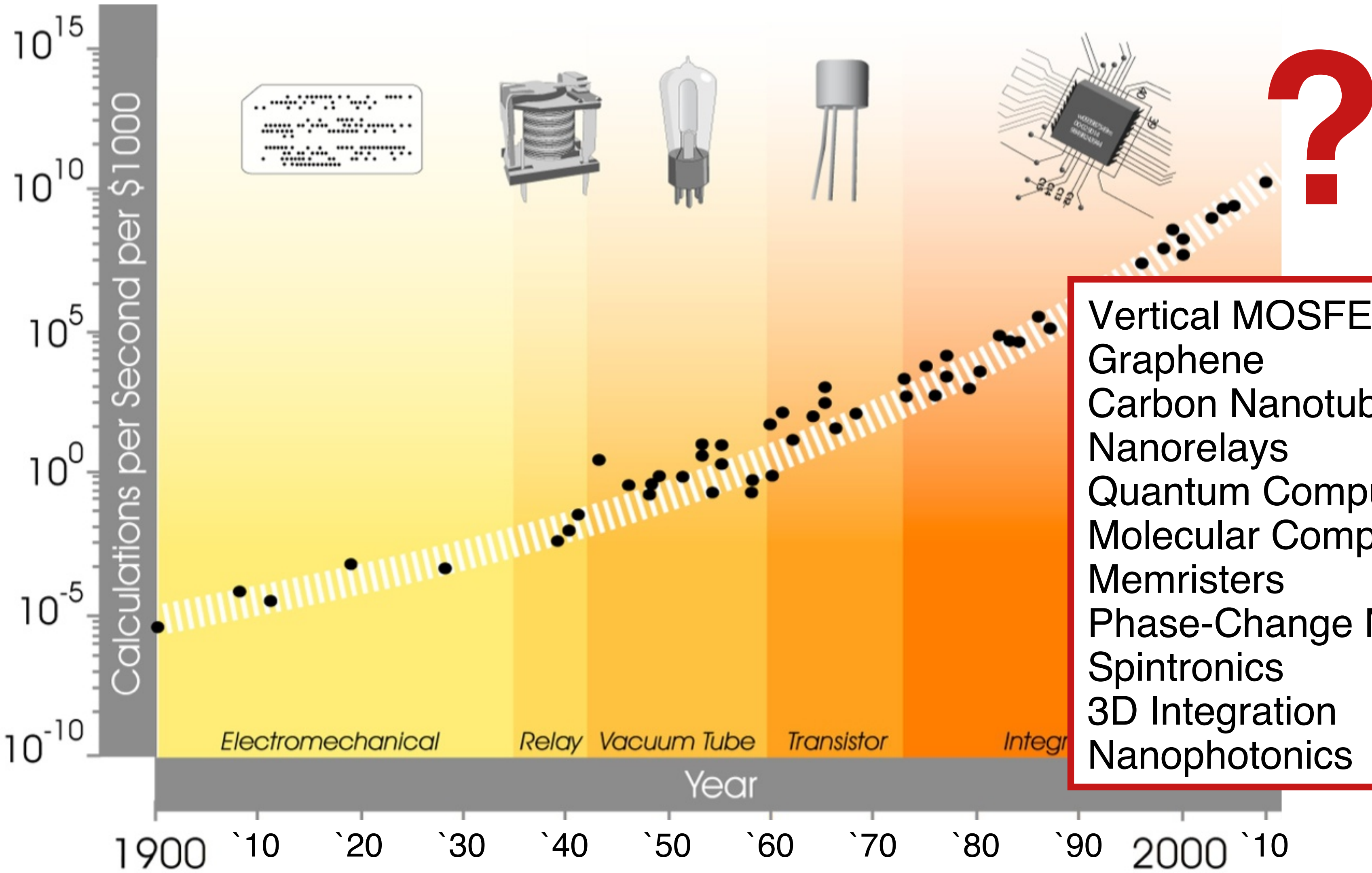
- Custom FPGA board for accelerating Bing search and other workloads
- Accelerators developed with/by app developers
- Tightly integrated into Microsoft data center's and cloud computing platforms, access gradually being given to outside developers

The Computer Systems Stack



Trend #3: Technology/Architecture Interface Changing Radically

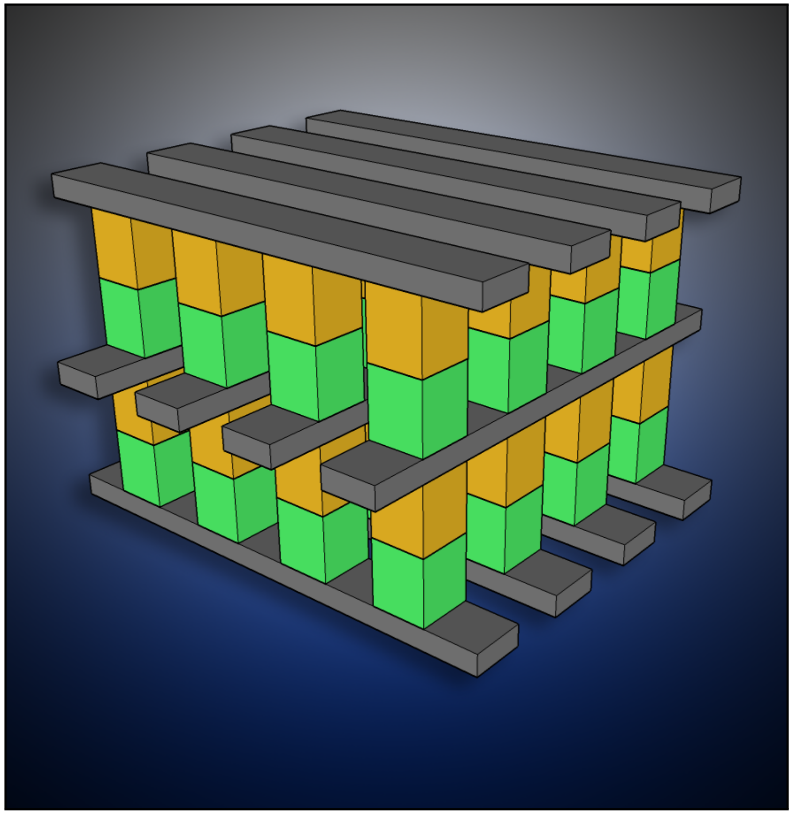
Trend 5: Emerging Device Technologies



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

Adapted from R. Kurzweil, “The Singularity is Near,” Penguin Books, 2006.

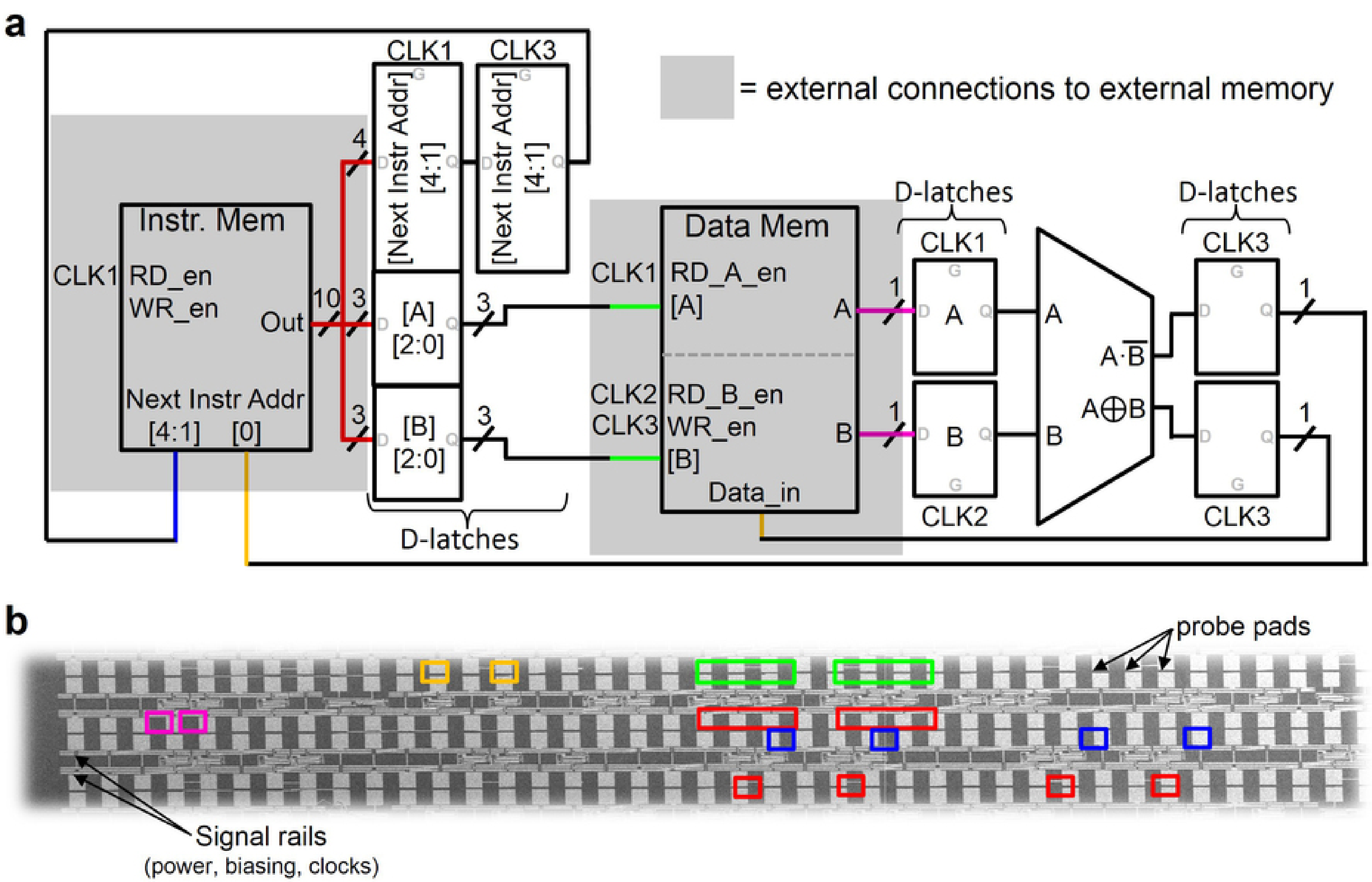
Examples of Emerging Technologies



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

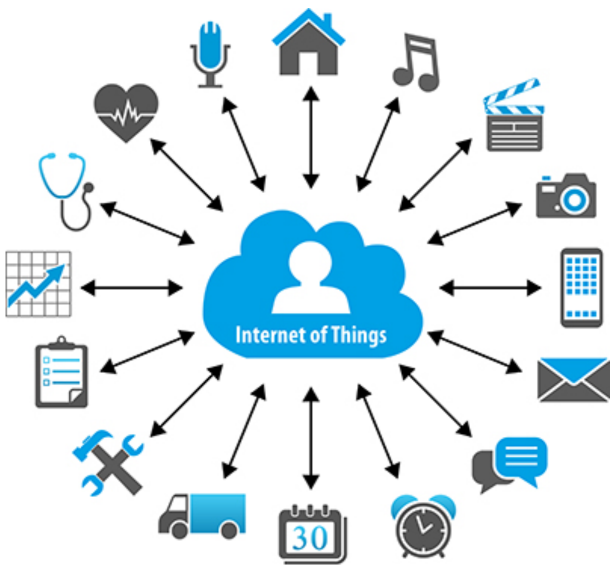
A Carbon Nanotube "Computer"



Adapted from M. Shulaker, et al., "Carbon Nanotube Computer," Nature, 2013.

Three Key Trends in Computer Engineering

Trend #1: Growing Diversity in Applications and Systems



- Application
- Algorithm
- PL
- OS
- ISA
- µArch
- RTL
- Gates
- Circuits
- Devices
- Technology

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

Application
Algorithm
PL
OS
ISA
µArch
RTL
Gates
Circuits
Devices
Technology

Agenda

The Computer Systems Stack

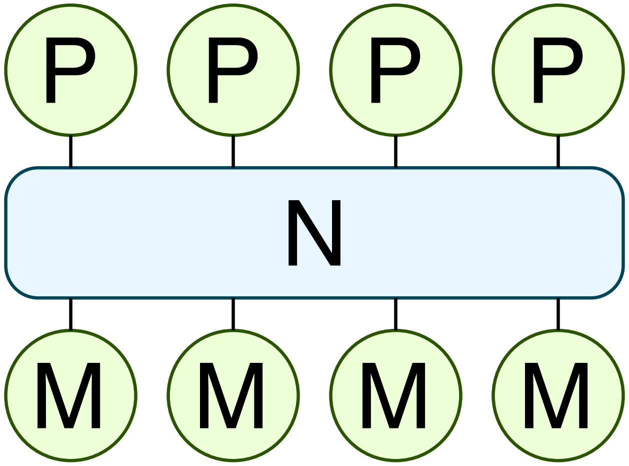
Activity 1

Trends in Computer Engineering

Activity 2

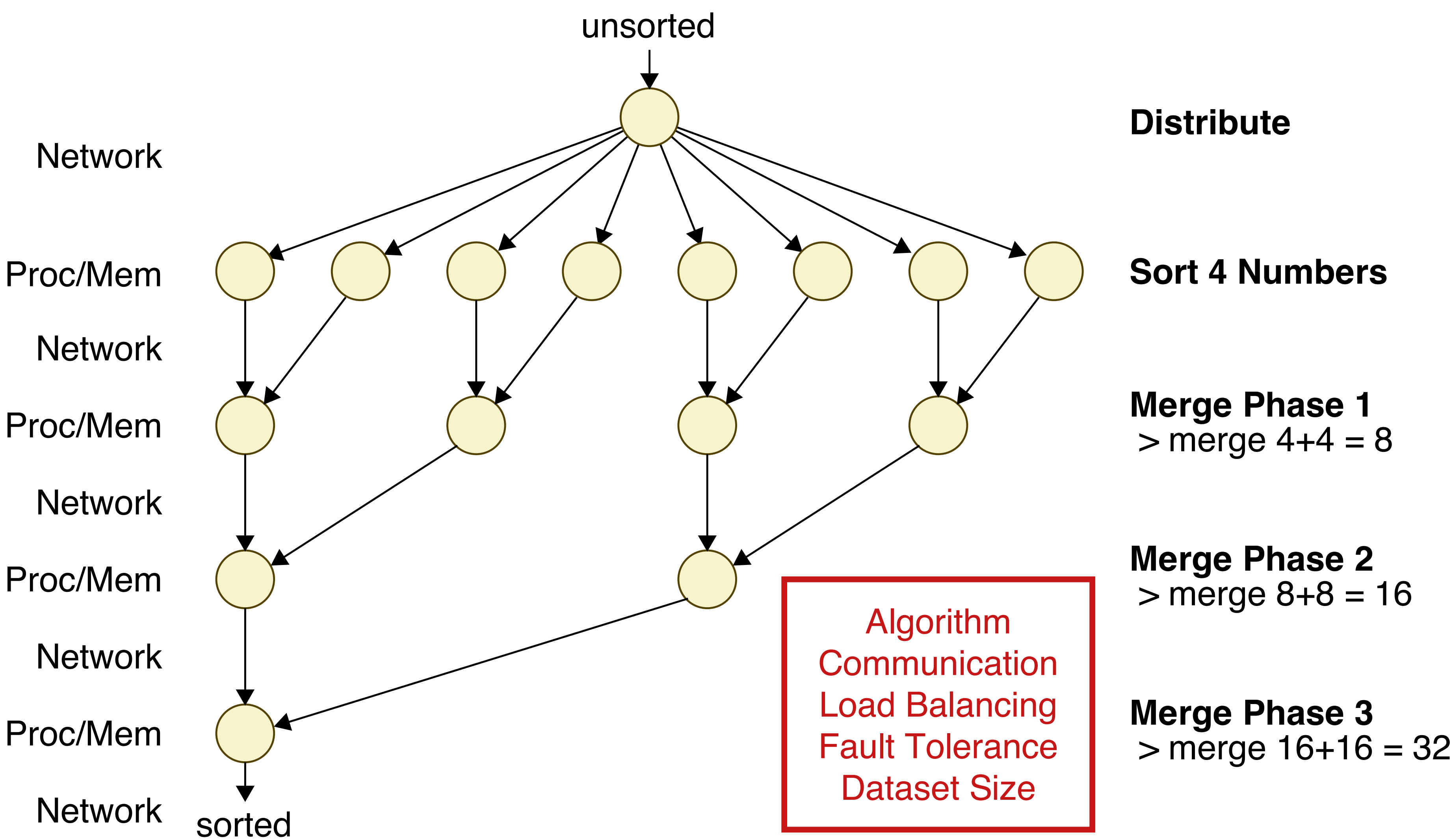
Hardware Acceleration for Deep Learning

Activity #2: Sorting with a Parallel Processor

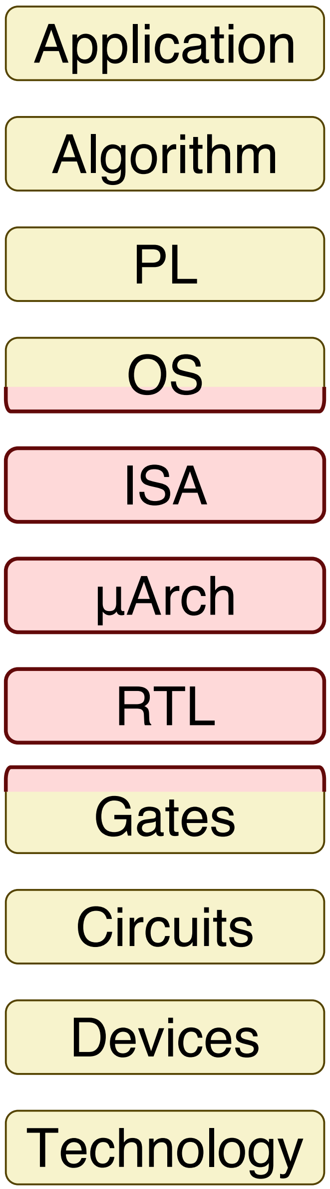


- **Application:** Sort 32 numbers
- **Simulated Parallel Computing System**
  - Processor: Group of 2–8 students
  - Memory: Worksheet, scratch paper
  - Network: Communicating between students
- **Activity Steps**
  - 1. Discuss strategy with group
  - 2. When instructor starts timer, master processor 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



Algorithm
Communication
Load Balancing
Fault Tolerance
Dataset Size



Agenda

The Computer Systems Stack

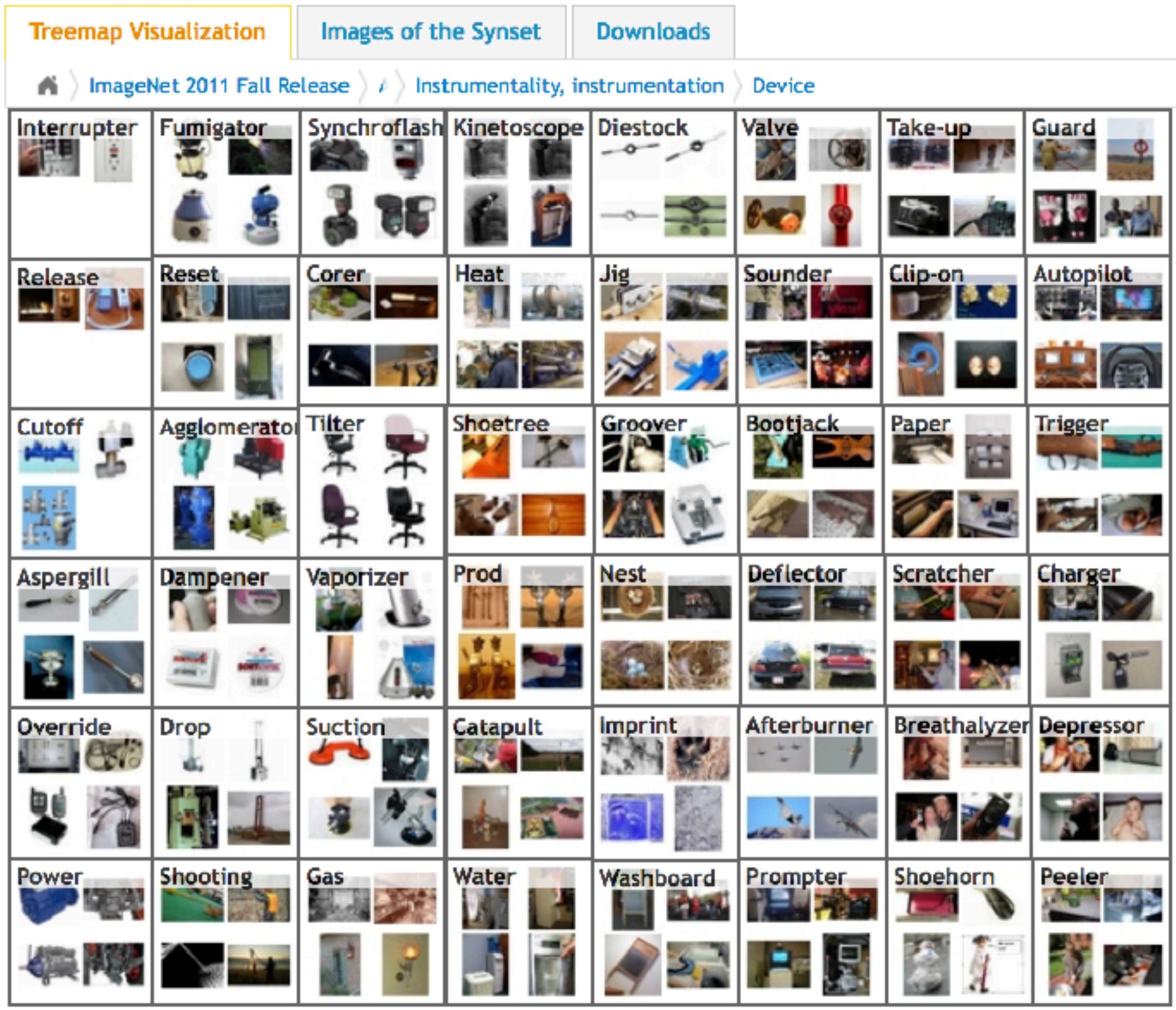
Activity 1

Trends in Computer Engineering

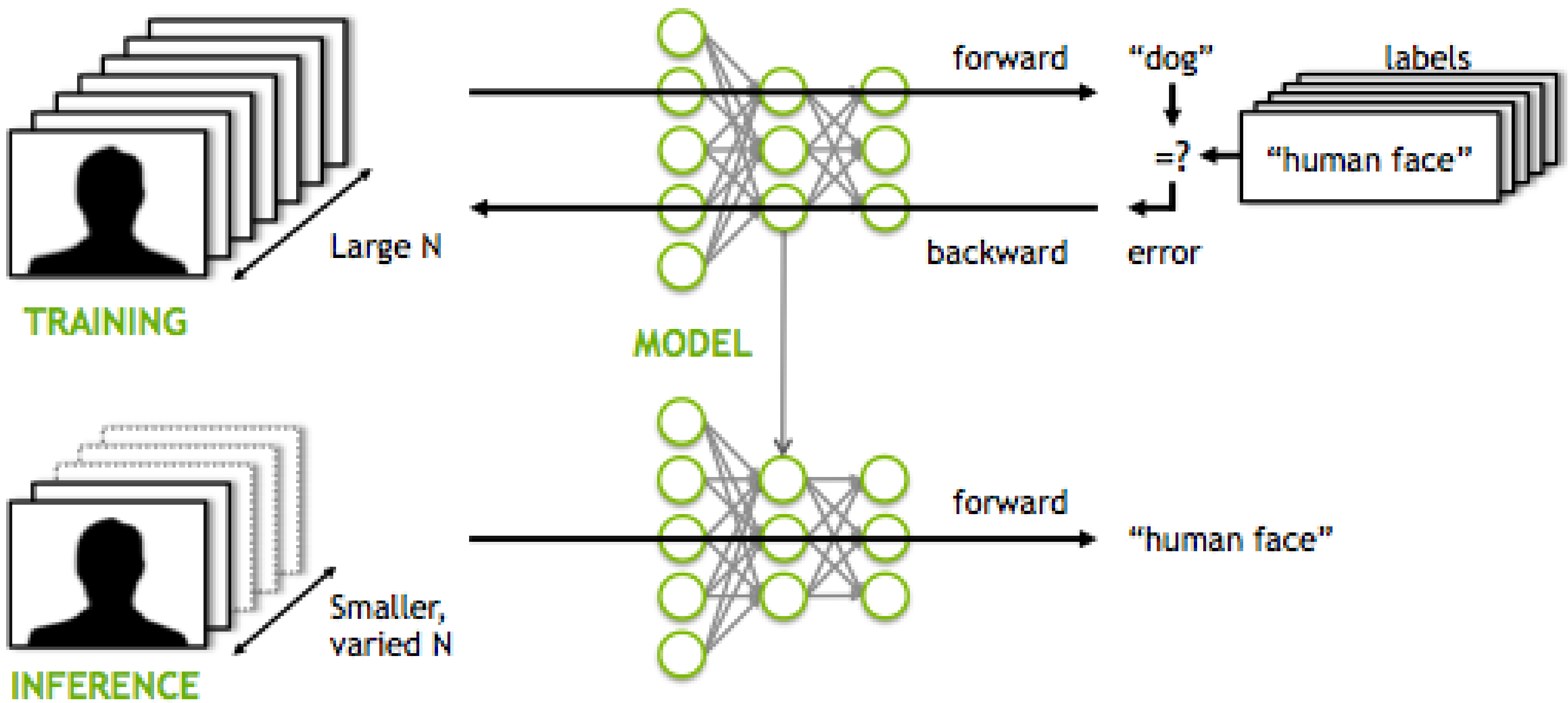
Activity 2

Hardware Acceleration for Deep Learning

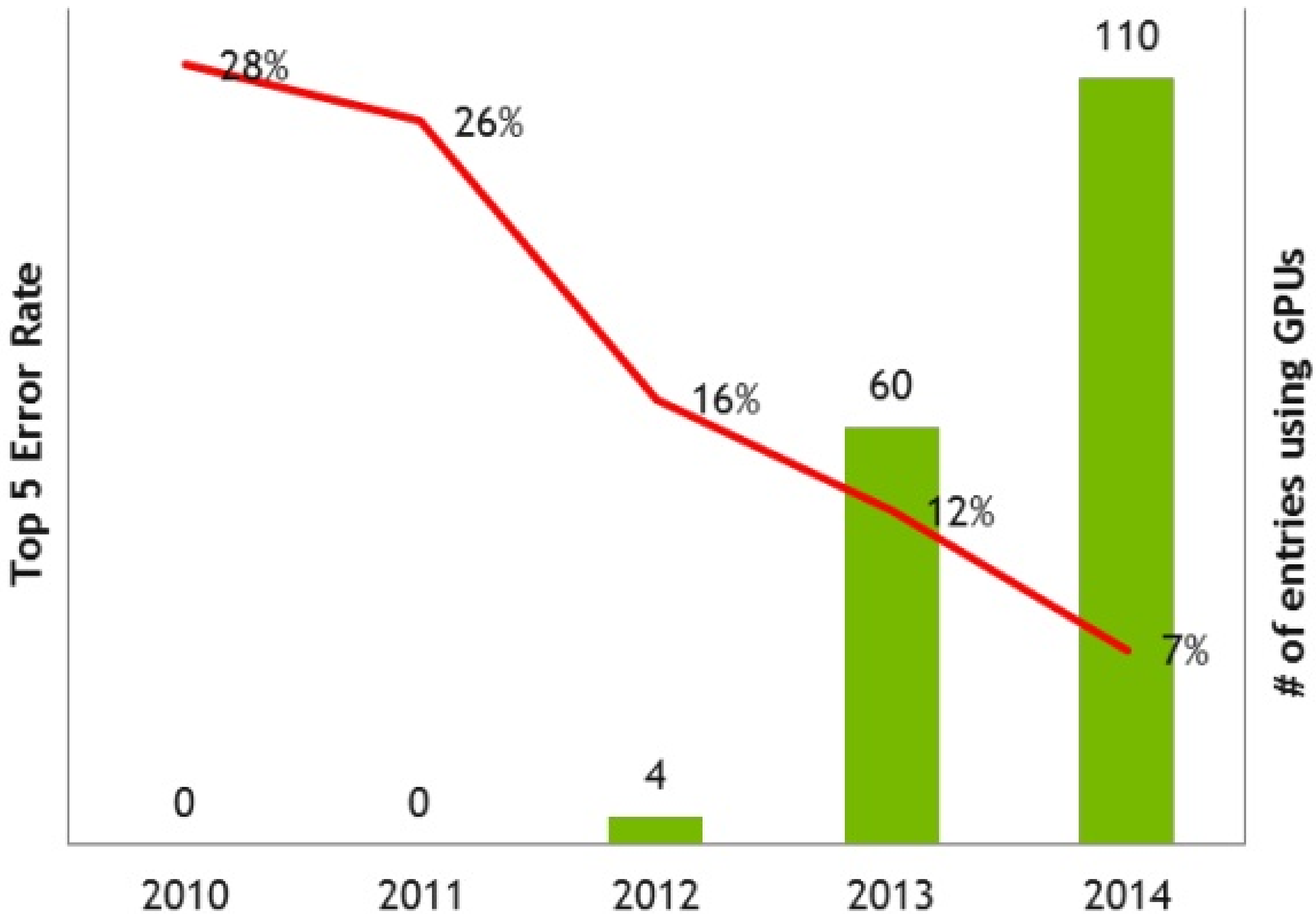
ImageNet: Object Recognition Competition



Machine Learning: Training vs. Inference

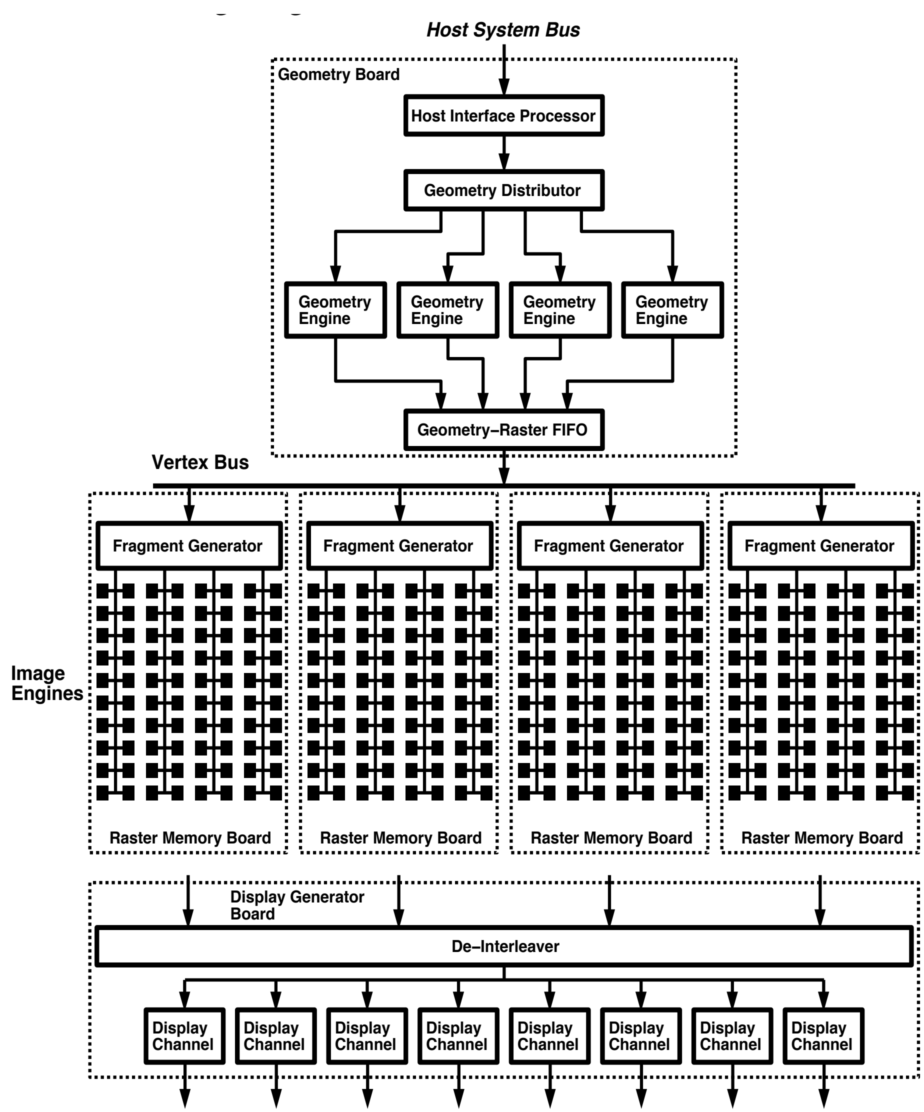


An Inflection Point due to Algorithms and Hardware



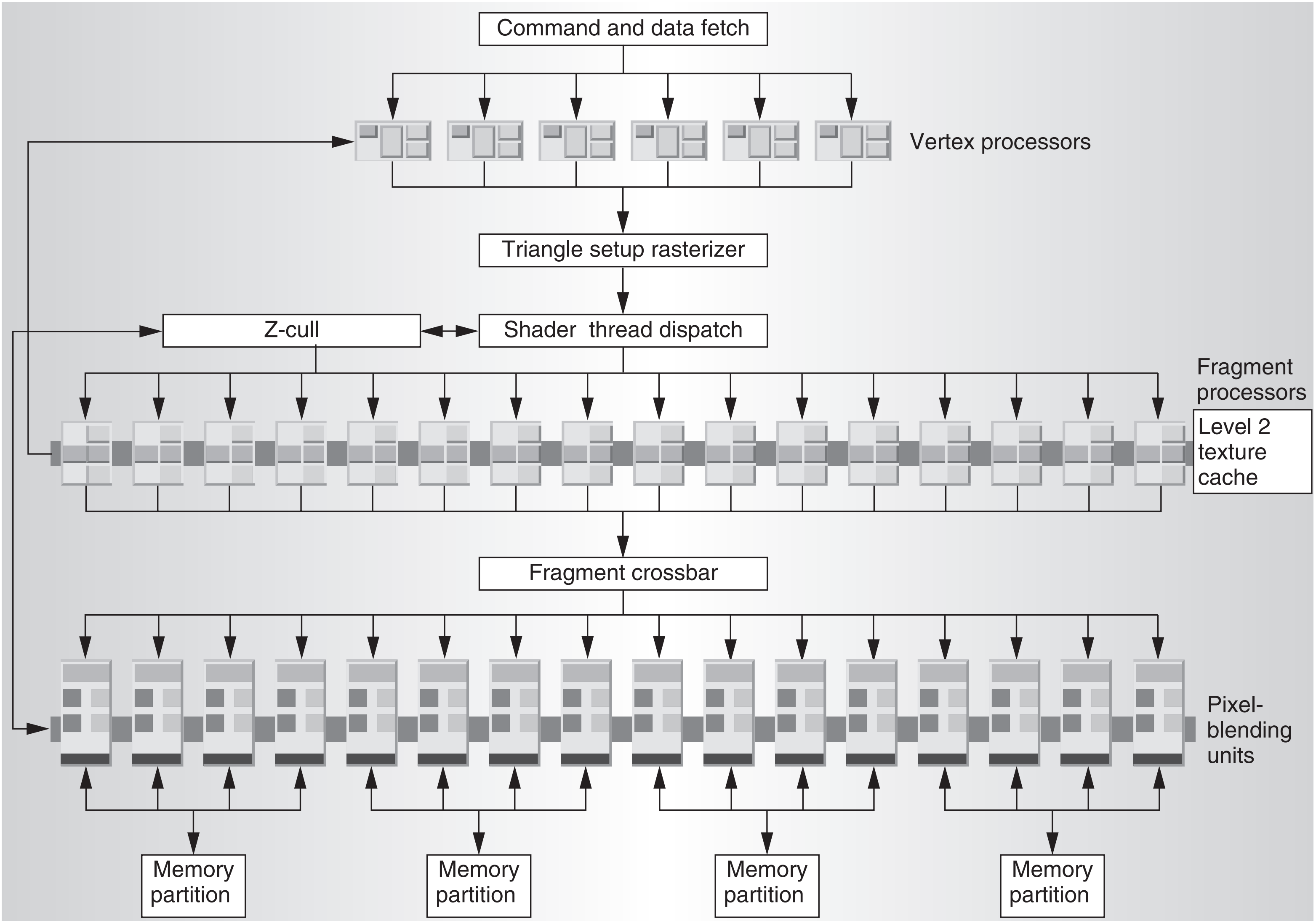
Adapted from A. Gray, “NVIDIA and IBM Cloud Support ImageNet Large Scale Visual Recognition Challenge,” NVIDIA Blog, 2015.

SGI InfiniteReality GPU



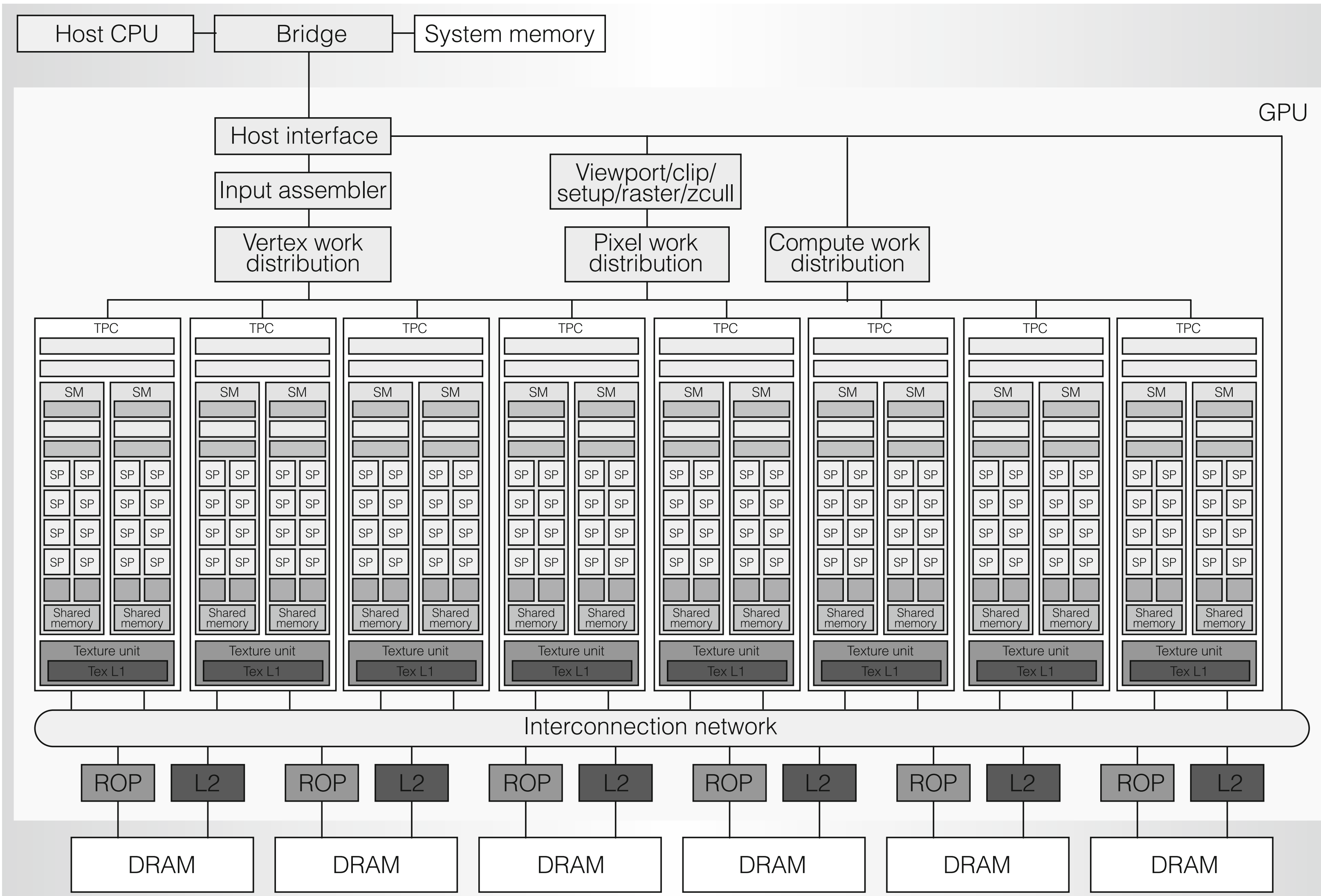
Adapted from J. Montrym, "InfiniteReality: A Real-Time Graphics System," ACM SIGGRAPH, 1997.

NVIDIA GeForce 6800



Adapted from J. Montrym, "The GeForce 6800," IEEE Micro, Mar/Apr 2005.

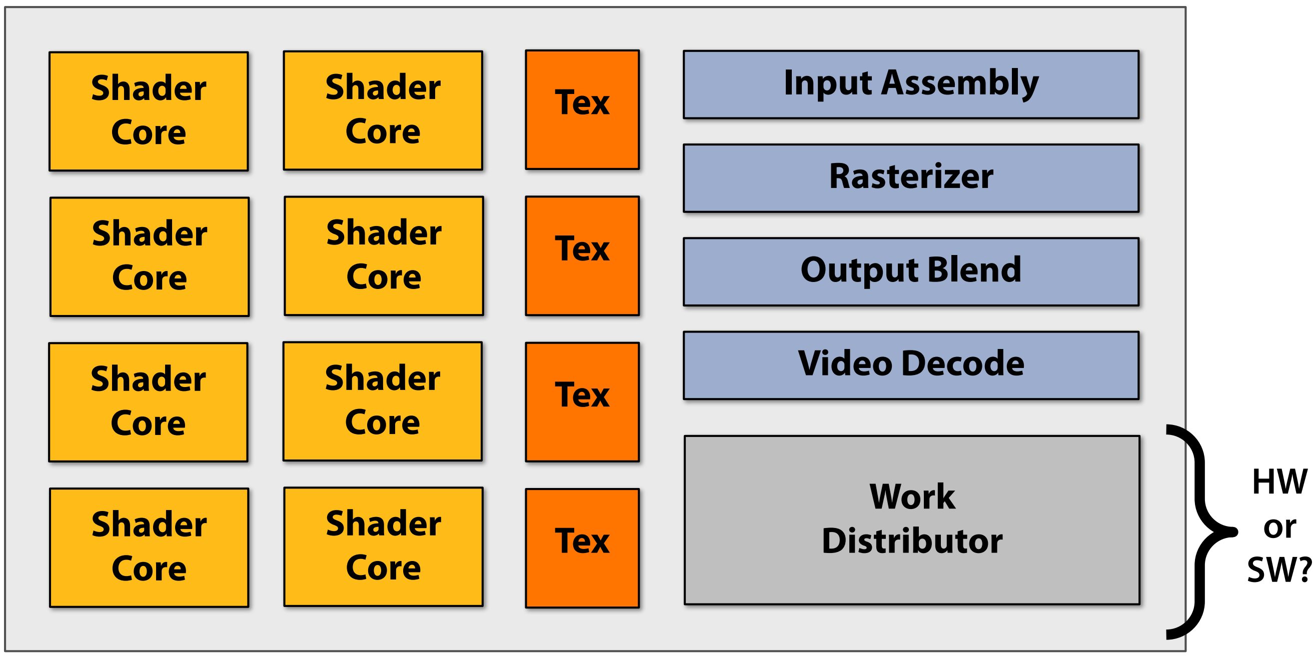
NVIDIA G80



Adapted from E. Lindholm, “NVIDIA Tesla: A Unified Graphics and Computing Architecture,” IEEE Micro, Mar/Apr 2008.

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

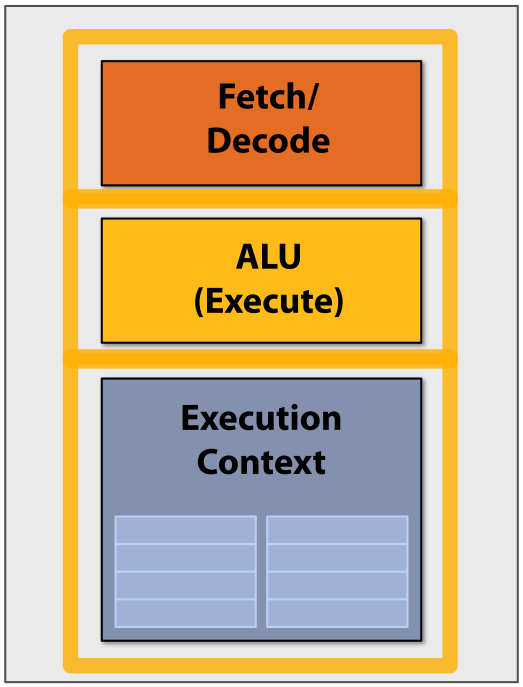
```
sampler mySamp;
Texture2D<float3> myTex;
float3 lightDir;

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);
}
```

```
<diffuseShader>:
sample r0, v4, t0, s0
mul  r3, v0, cb0[0]
madd r3, v1, cb0[1], r3
madd r3, v2, cb0[2], r3
clmp r3, r3, l(0.0), l(1.0)
mul  o0, r0, r3
mul  o1, r1, r3
mul  o2, r2, r3
mov  o3, l(1.0)
```

1 shaded fragment output record

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

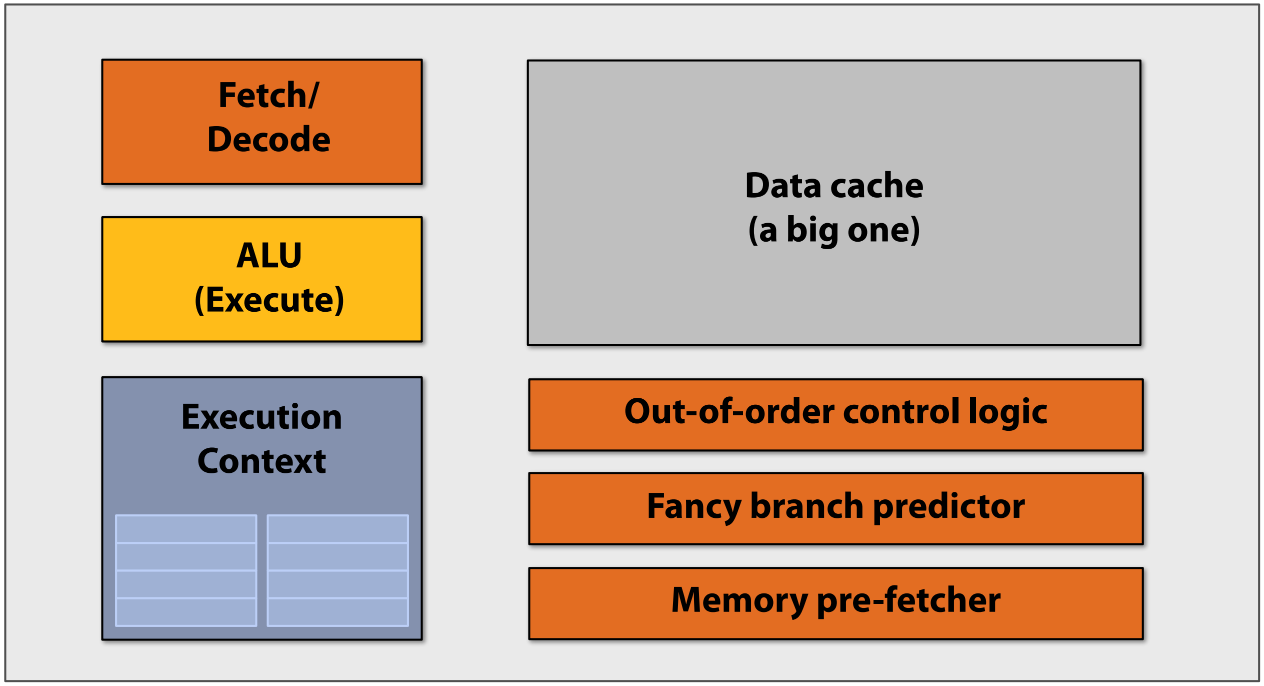
Executing a Shader




```
<diffuseShader>:
sample r0, v4, t0, s0
mul  r3, v0, cb0[0]
madd r3, v1, cb0[1], r3
madd r3, v2, cb0[2], r3
clmp r3, r3, l(0.0), l(1.0)
mul  o0, r0, r3
mul  o1, r1, r3
mul  o2, r2, r3
mov  o3, l(1.0)
```

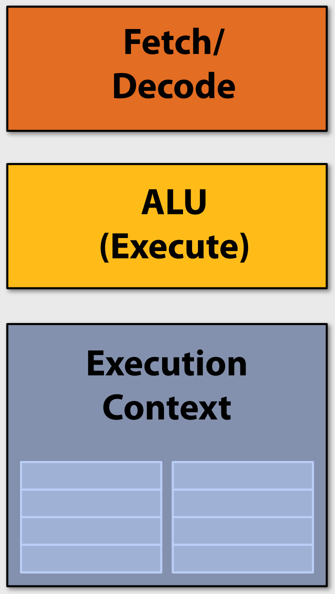
Adapted from K. Fatahalian, “Beyond Programmable Shading Course,” ACM SIGGRAPH 2010.

"CPU-Style" Cores



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

Slimming Down

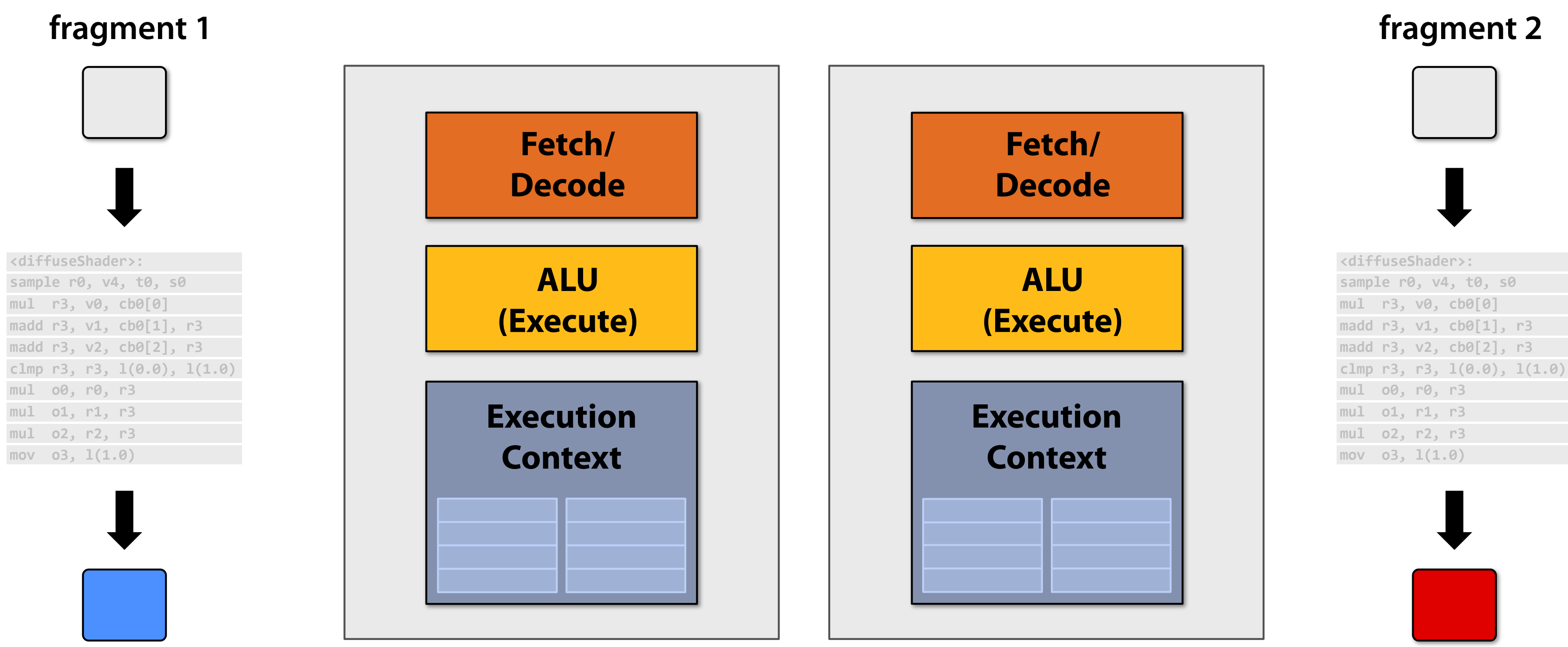


Idea #1:

Remove components that help a single instruction stream run fast

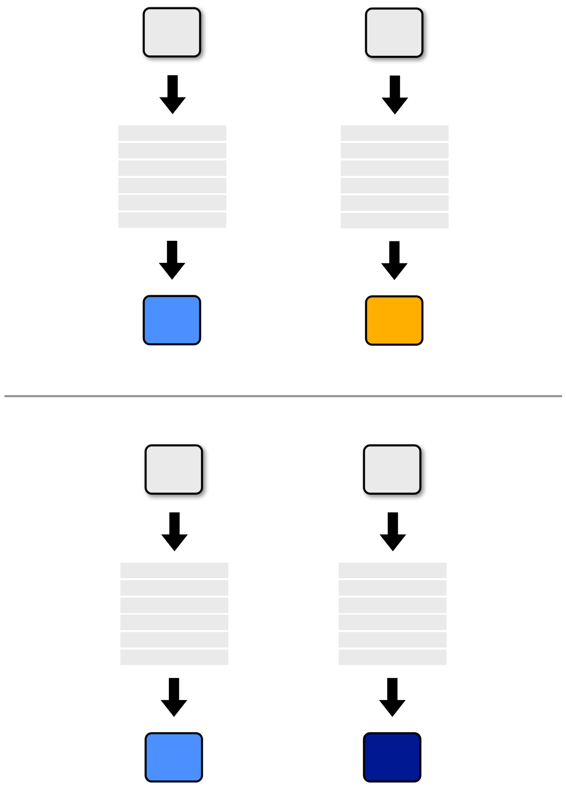
Adapted from K. Fatahalian, "Beyond Programmable Shading Course," ACM SIGGRAPH 2010.

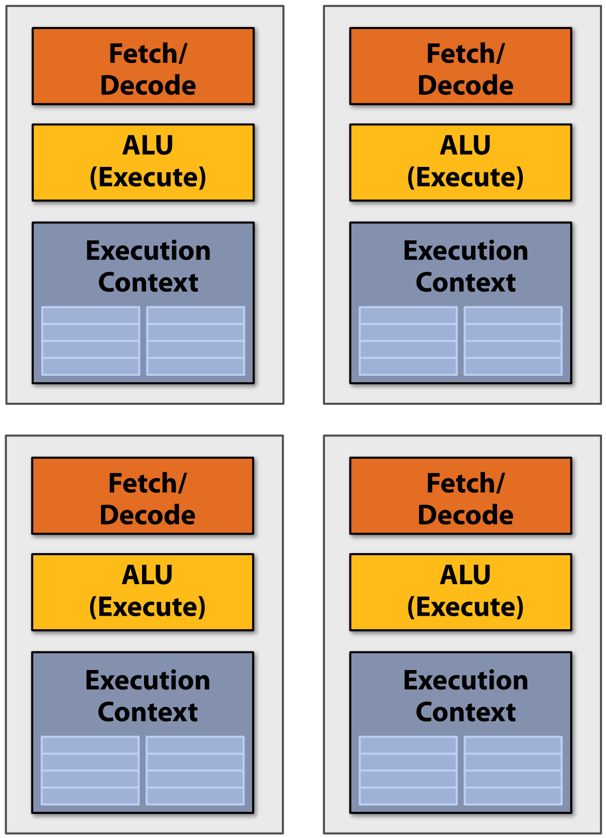
Two Cores (Execute Two Fragments in Parallel)



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

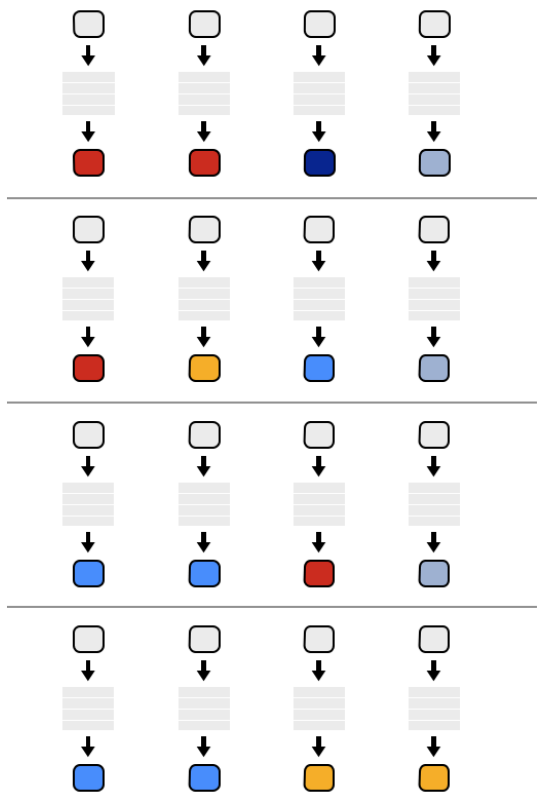
Four Cores (Execute Four Fragments in Parallel)

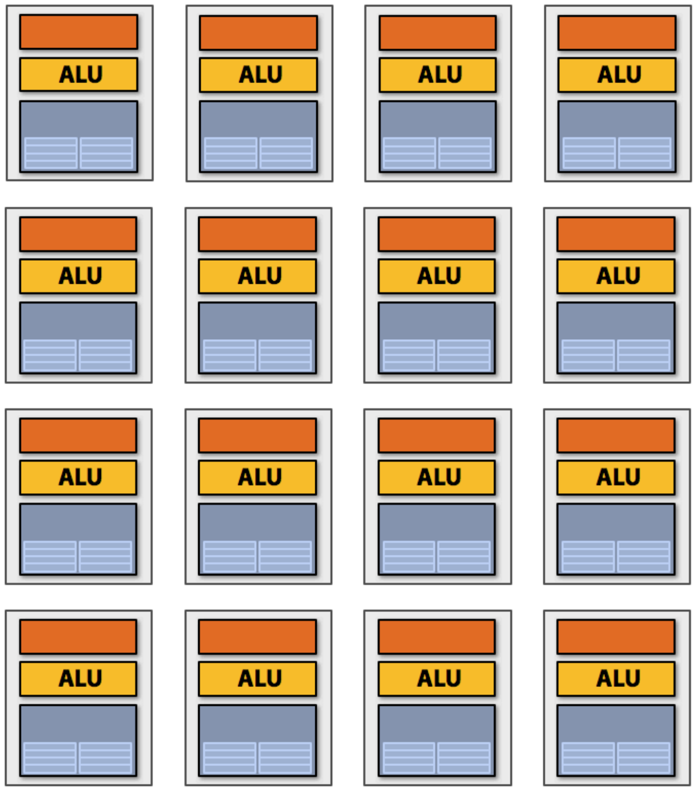




Adapted from K. Fatahalian, "Beyond Programmable Shading Course," ACM SIGGRAPH 2010.

16 Cores (Execute 16 Fragments in Parallel)

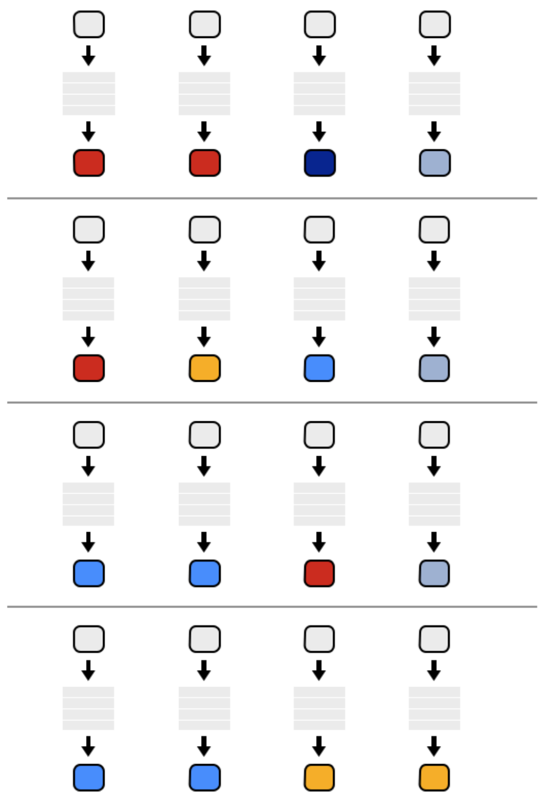




16 cores = 16 simultaneous instruction streams

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

Instruction Stream Sharing

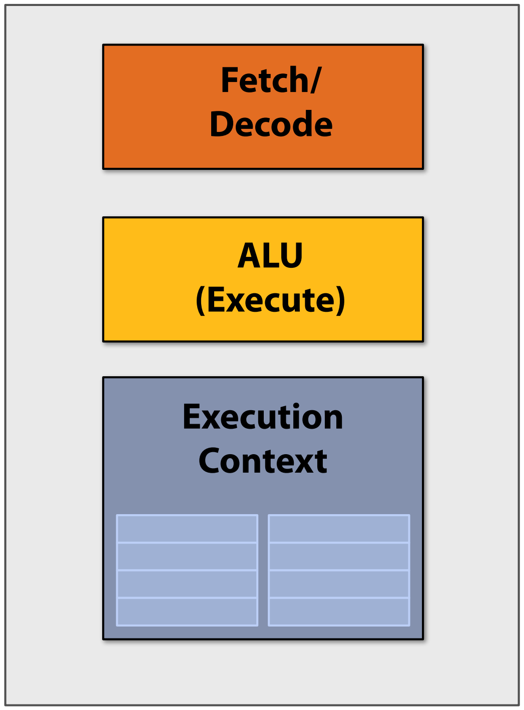


But ... many fragments should be able to share an instruction stream!

```
<diffuseShader>:
sample r0, v4, t0, s0
mul  r3, v0, cb0[0]
madd r3, v1, cb0[1], r3
madd r3, v2, cb0[2], r3
clmp r3, r3, l(0.0), l(1.0)
mul  o0, r0, r3
mul  o1, r1, r3
mul  o2, r2, r3
mov  o3, l(1.0)
```

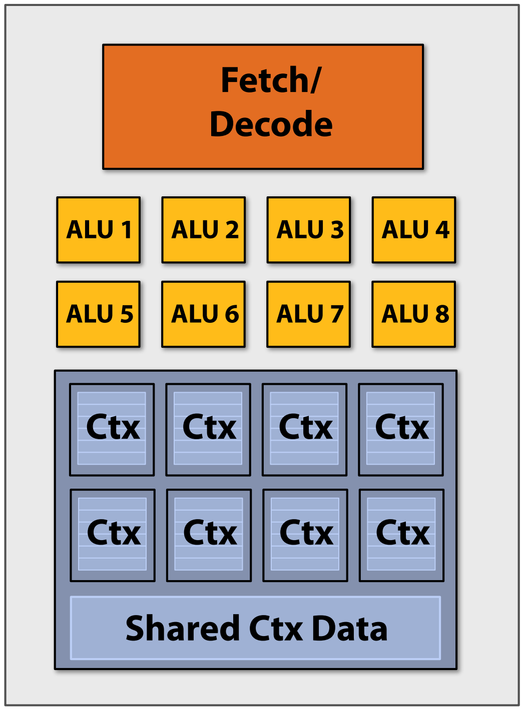
Adapted from K. Fatahalian, “Beyond Programmable Shading Course,” ACM SIGGRAPH 2010.

Recall: Simple Processing Core



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

Add ALUs

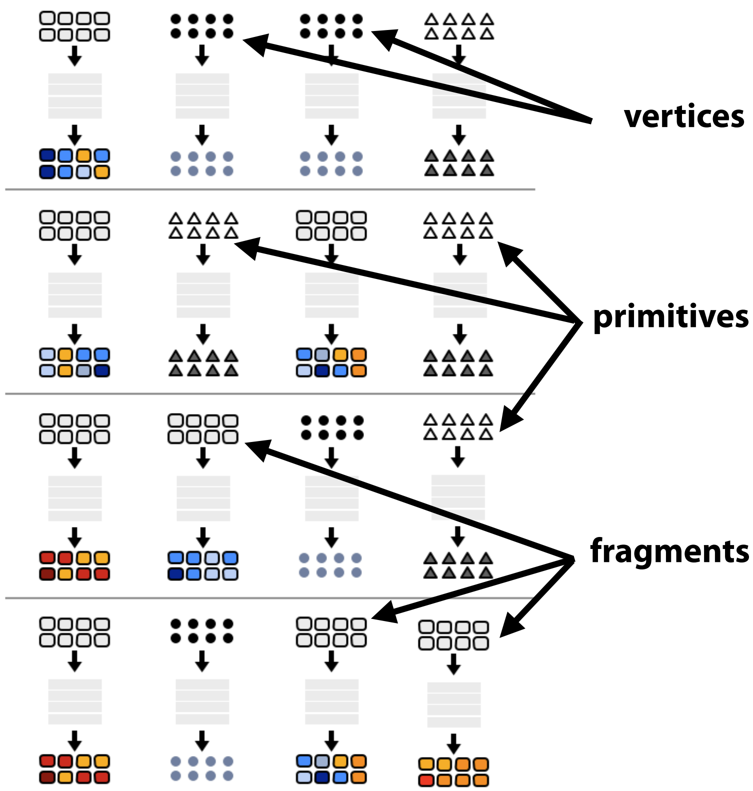


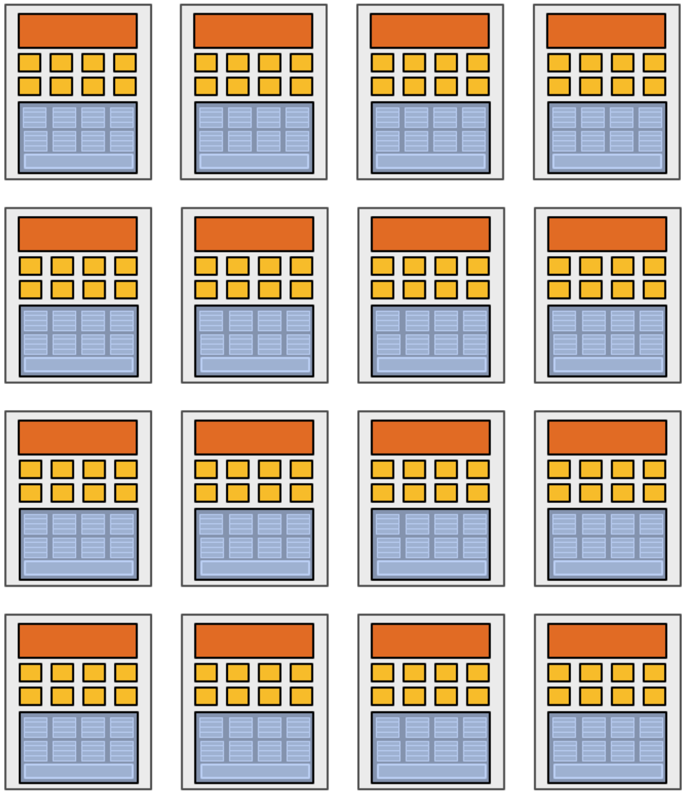
Idea #2:
Amortize cost/complexity of managing an instruction stream across many ALUs

SIMD processing

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

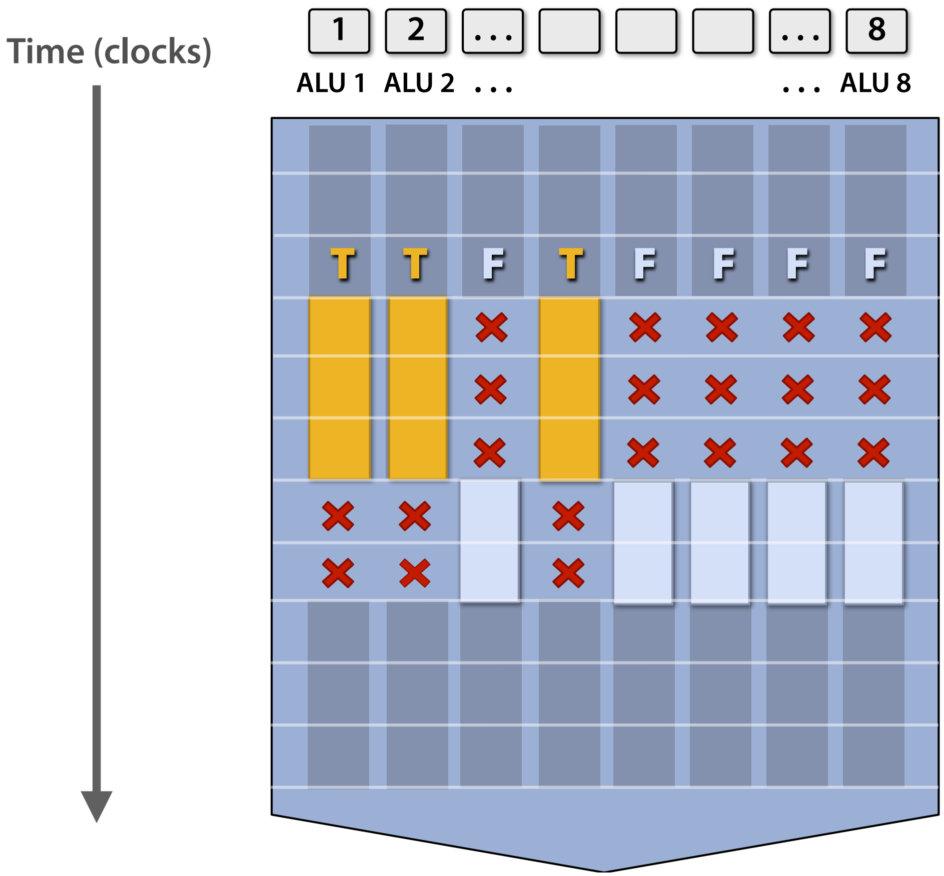
Execute 128 Fragments in Parallel





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

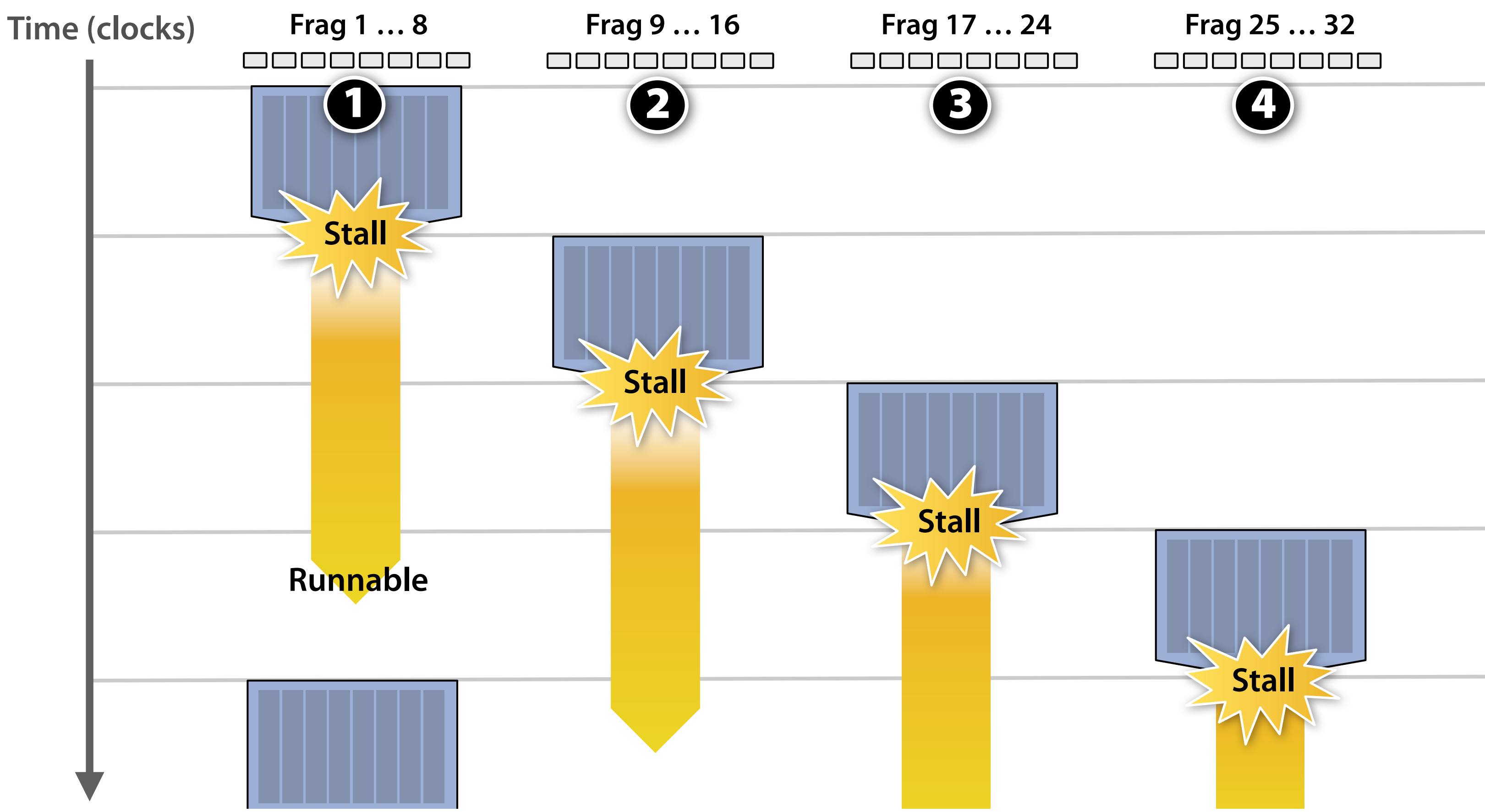
But What About Branches?




```
<unconditional
 shader code>

if (x > 0) {
    y = pow(x, exp);
    y *= Ks;
    refl = y + Ka;
} else {
    x = 0;
    refl = Ka;
}

<resume unconditional
 shader code>
```

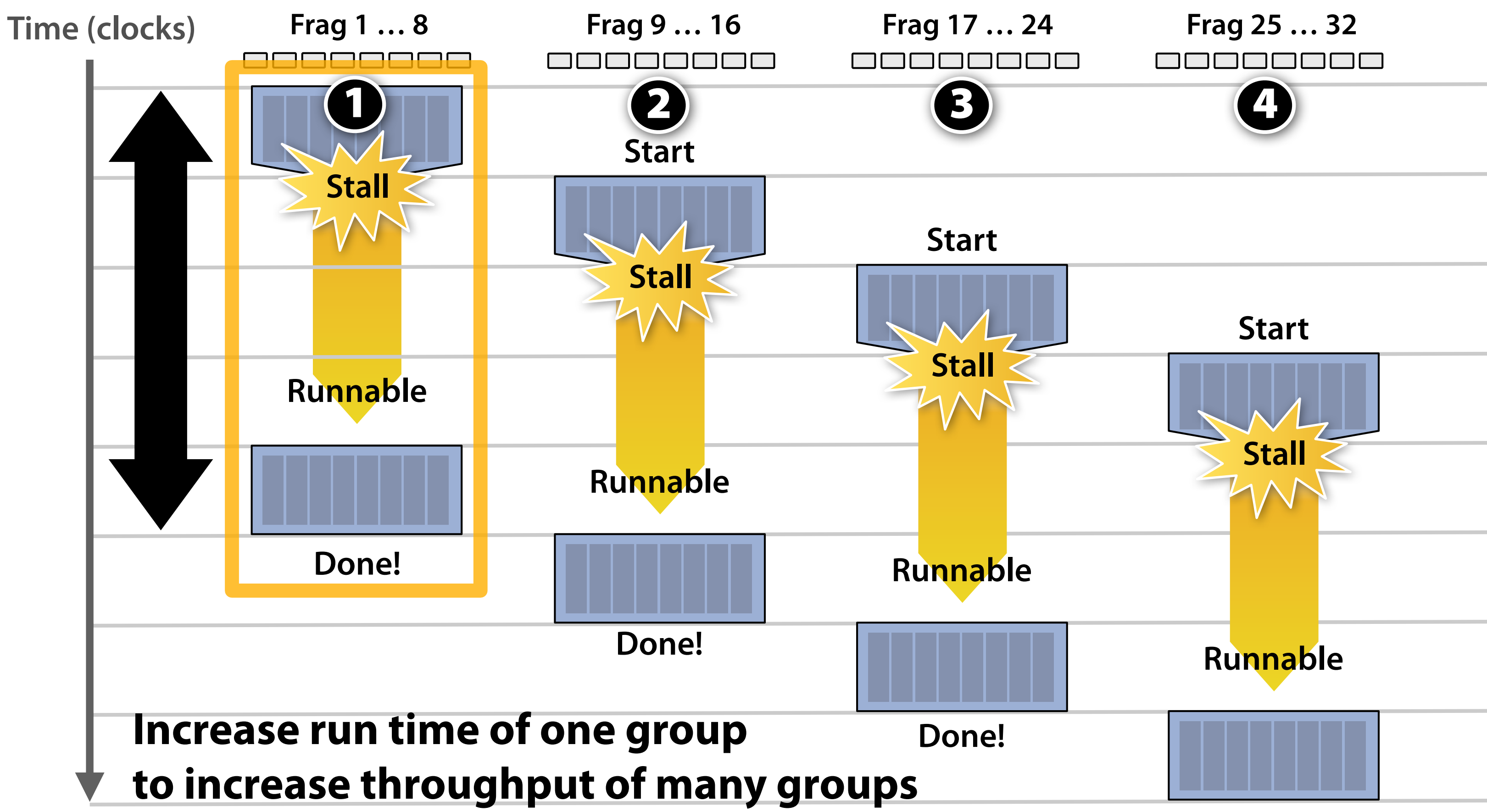
Adapted from K. Fatahalian, “Beyond Programmable Shading Course,” ACM SIGGRAPH 2010.

But What About Memory Stalls?



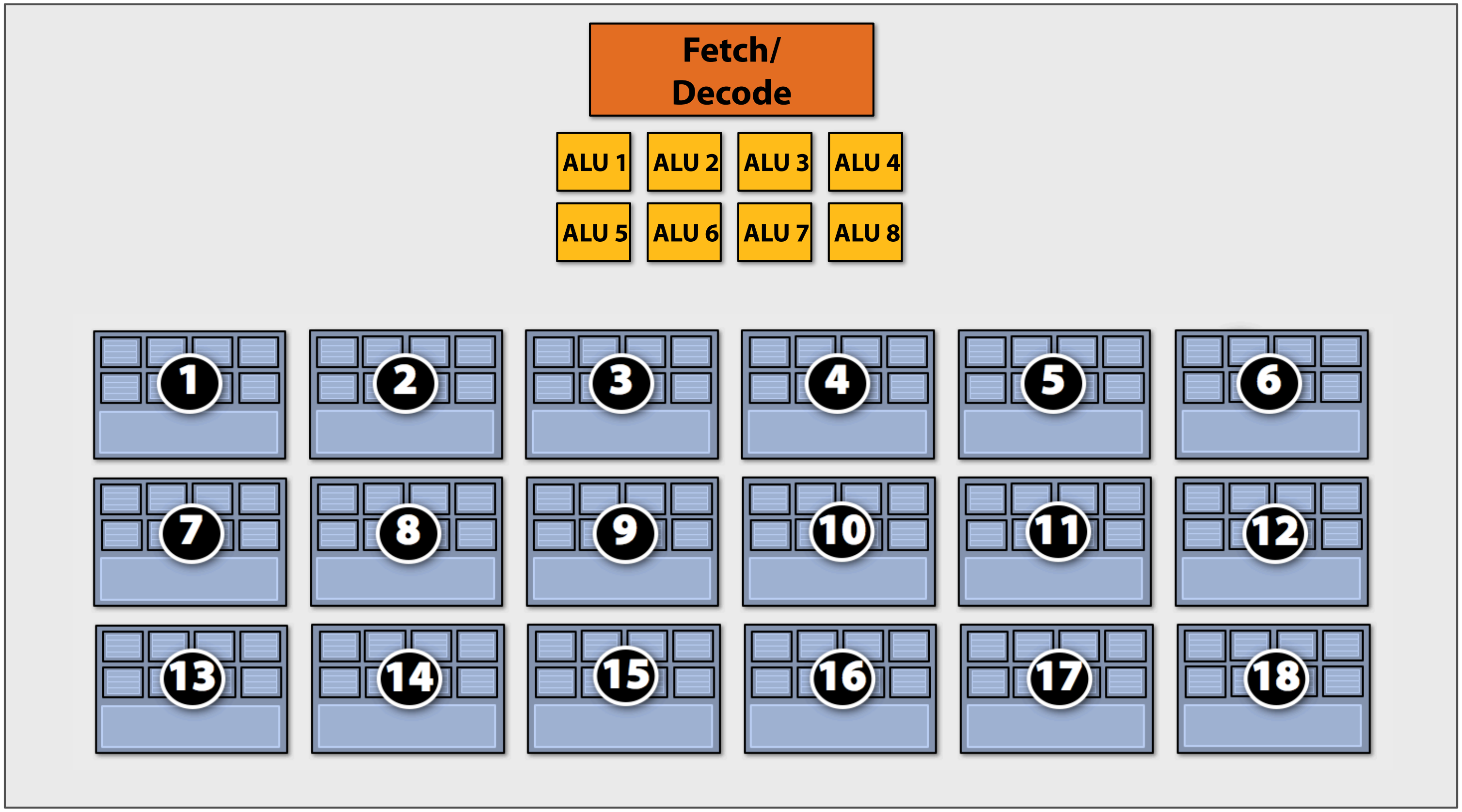
Adapted from K. Fatahalian, "Beyond Programmable Shading Course," ACM SIGGRAPH 2010.

Key is Throughput!



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

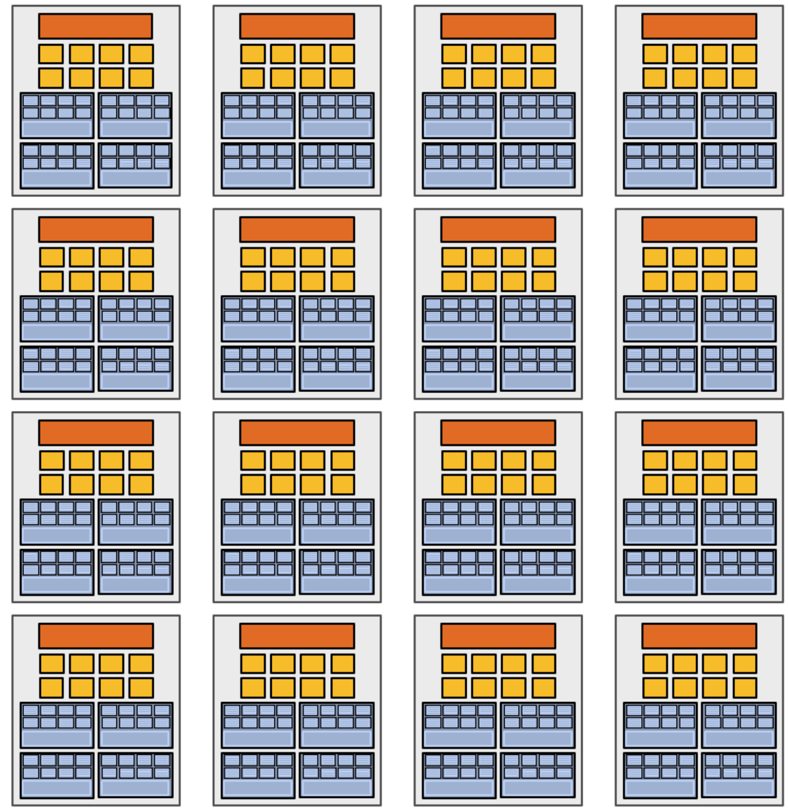
8 mul-add ALUs per core
(128 total)

16 simultaneous
instruction streams

64 concurrent (but interleaved)
instruction streams

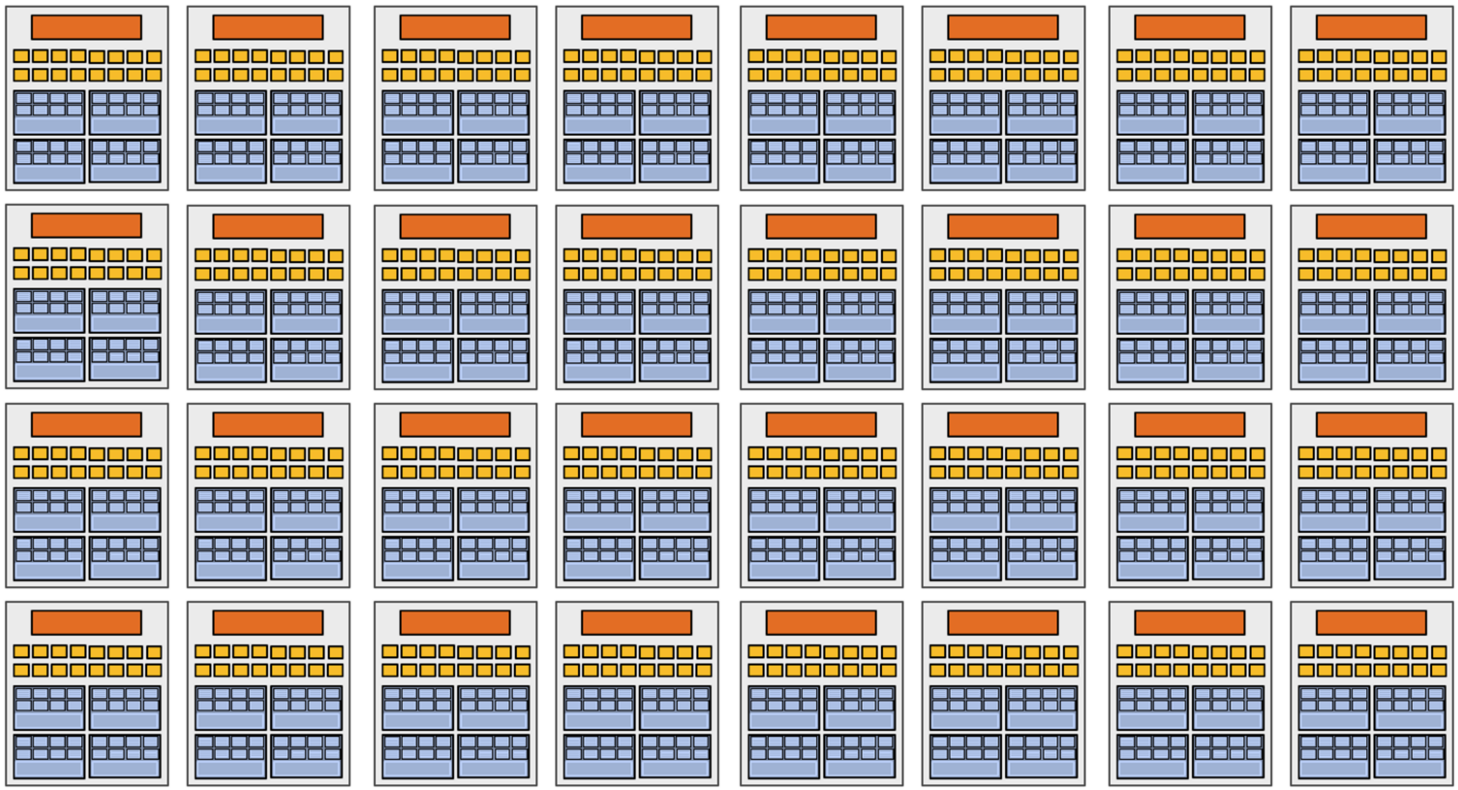
512 concurrent fragments

= 256 GFLOPs (@ 1GHz)



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

Complete “Big” GPU



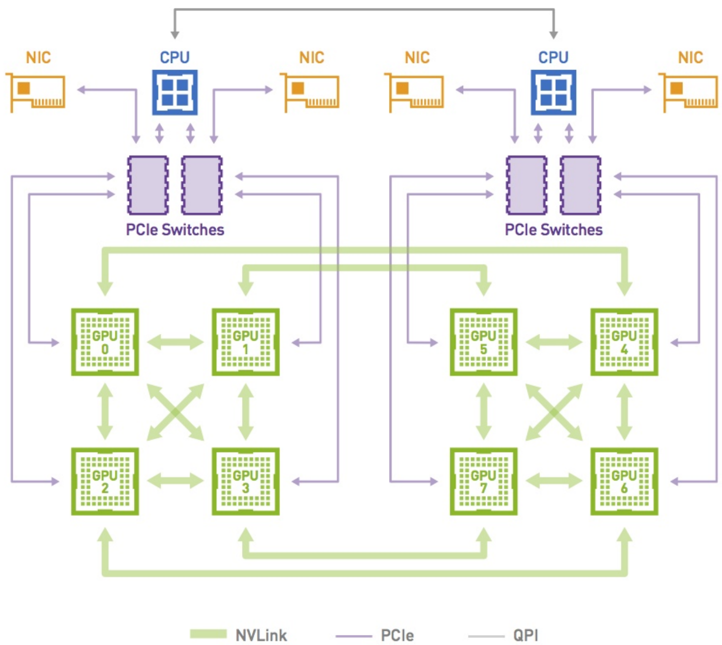
32 cores, 16 ALUs per core (512 total) = 1 TFLOP (@ 1 GHz)

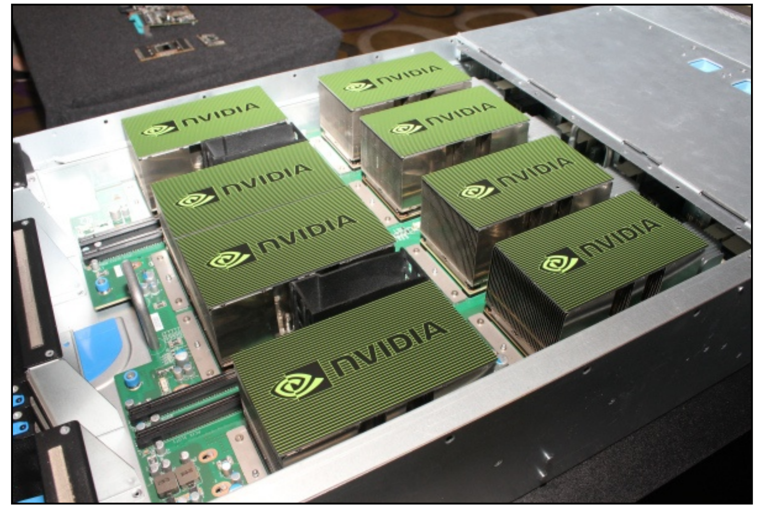
Adapted from K. Fatahalian, “Beyond Programmable Shading Course,” ACM SIGGRAPH 2010.

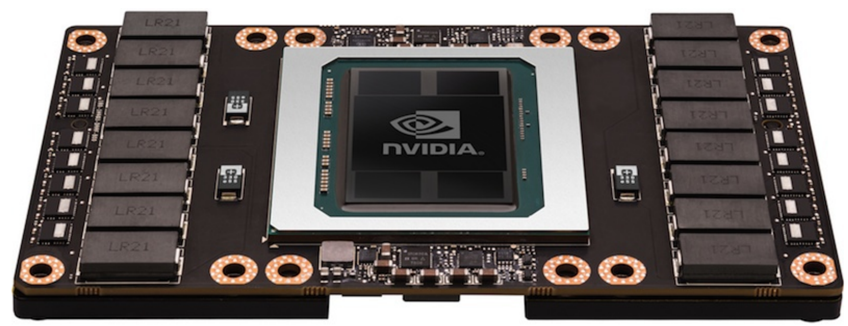
Using GPUs for General-Purpose Computing

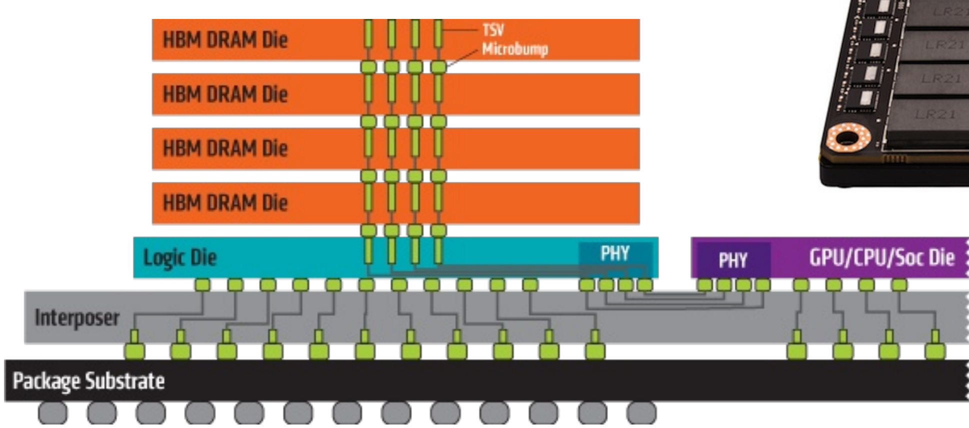
```
int main( int argc, char* argv[] )
{
  // ... copy data to GPGPU ...
  vvadd<<<block_count,threads_per_block >>>
    ( dest, src0, src1, n );
  // ... copy data from GPGPU to CPU ...
}

__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

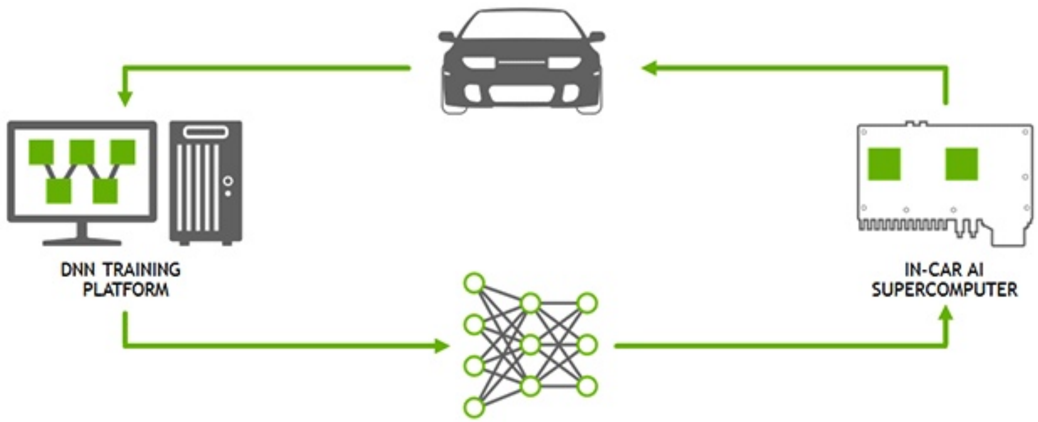




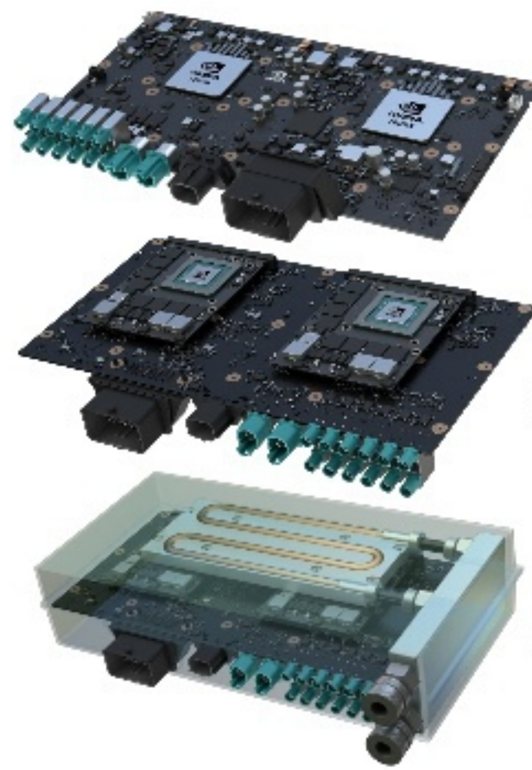




NVIDIA PX 2 for Deep Learning Inference



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

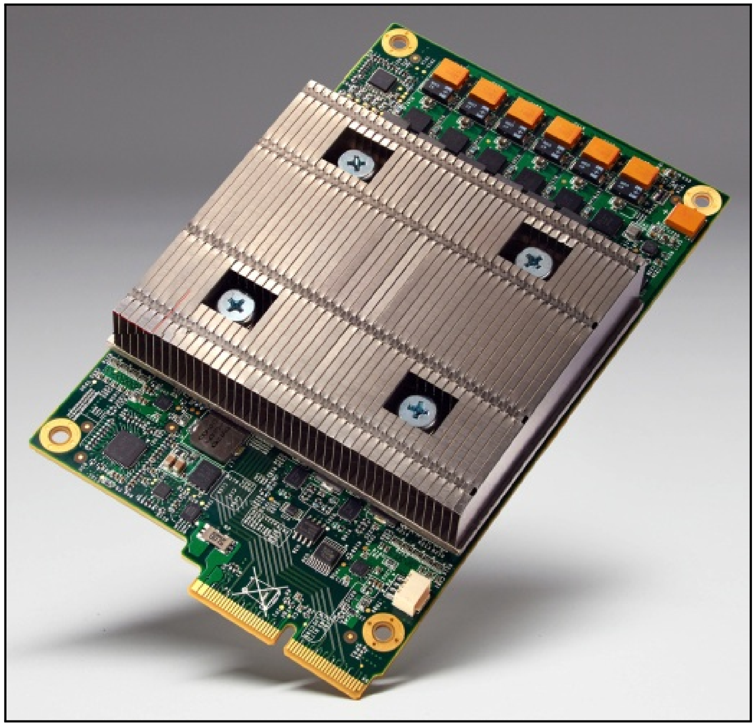


Dual Tegras on Top

Dual Discrete GPUs on the Bottom

Liquid Cooled if All Devices used

Hardware Xcel for ML is Significant Growth Area

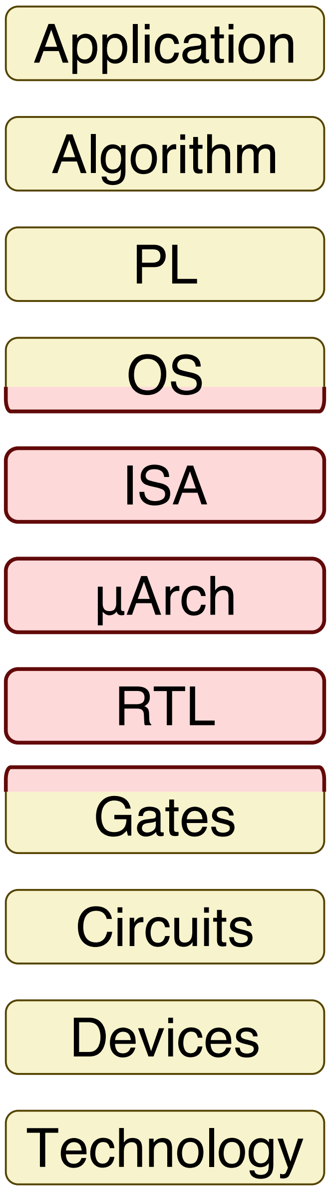


Google's TPU

- Custom chip for accelerating Google's TensorFlow library
- Tightly integrated into Google's data centers

Hardware ML Startups

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



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

```
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