

Preparing for Extreme Heterogeneity in High Performance Computing

Jeffrey S. Vetter

With many contributions from FTG Group and Colleagues

R-CCS International Symposium
Kobe
18 Feb 2020



ORNL is managed by UT-Battelle, LLC for the US Department of Energy



<http://ft.ornl.gov>

vetter@computer.org

Highlights

- Recent trends in extreme-scale HPC paint an ambiguous future
 - Contemporary systems provide evidence that power constraints are driving architectures to change rapidly
 - Multiple architectural dimensions are being (dramatically) redesigned: Processors, node design, memory systems, I/O
 - Complexity is our main challenge
- Applications and software systems are all reaching a state of crisis
 - Applications will not be functionally or performance portable across architectures
 - Programming and operating systems need major redesign to address these architectural changes
 - Procurements, acceptance testing, and operations of today's new platforms depend on performance prediction and benchmarking.
- We need portable programming models and performance prediction now more than ever!
 - Heterogeneous processing
 - **OpenACC->FGPAs**
 - **Intelligent runtime system (IRIS)**
 - Clacc – OpenACC support in LLVM (not covered today)
 - OpenACC dialect of MLIR for Flang Fortran (not covered today)
 - Emerging memory hierarchies (NVM)
 - DRAGON – transparent NVM access from GPUs (not covered today)
 - NVL-C – user management of nonvolatile memory in C (not covered today)
 - Papyrus – parallel aggregate persistent storage (not covered today)
- Performance prediction is critical for design and optimization (not covered today)

Time for a short poll...

History

Q: Think back 10 years. How many of you would have predicted that many of our top HPC systems would be GPU-based architectures?

Yes

No

Revisionists 😊

Future

Q: Think forward 10 years. How many of you predict that most of our top HPC systems will have the following architectural features?

Assume general purpose multicore CPU

GPU

FPGA/Reconfigurable processor

Neuromorphic processor

Deep learning processor

Quantum processor

RISC-V processor

Some new unknown processor

All/some of the above in one SoC

Implications

Q: Now, imagine you are building a new application with an expected ~3M LOC and 20 team members over the next 10 years. What on-node programming model/system do you use?

C, C++ XX, Fortran XX

Metaprogramming, etc (e.g., AMP, Kokkos, RAJA, SYCL)

CUDA, cu***, HIP, OpenCL

Directives: OpenMP XX, OpenACC XX

R, Python, Matlab, etc

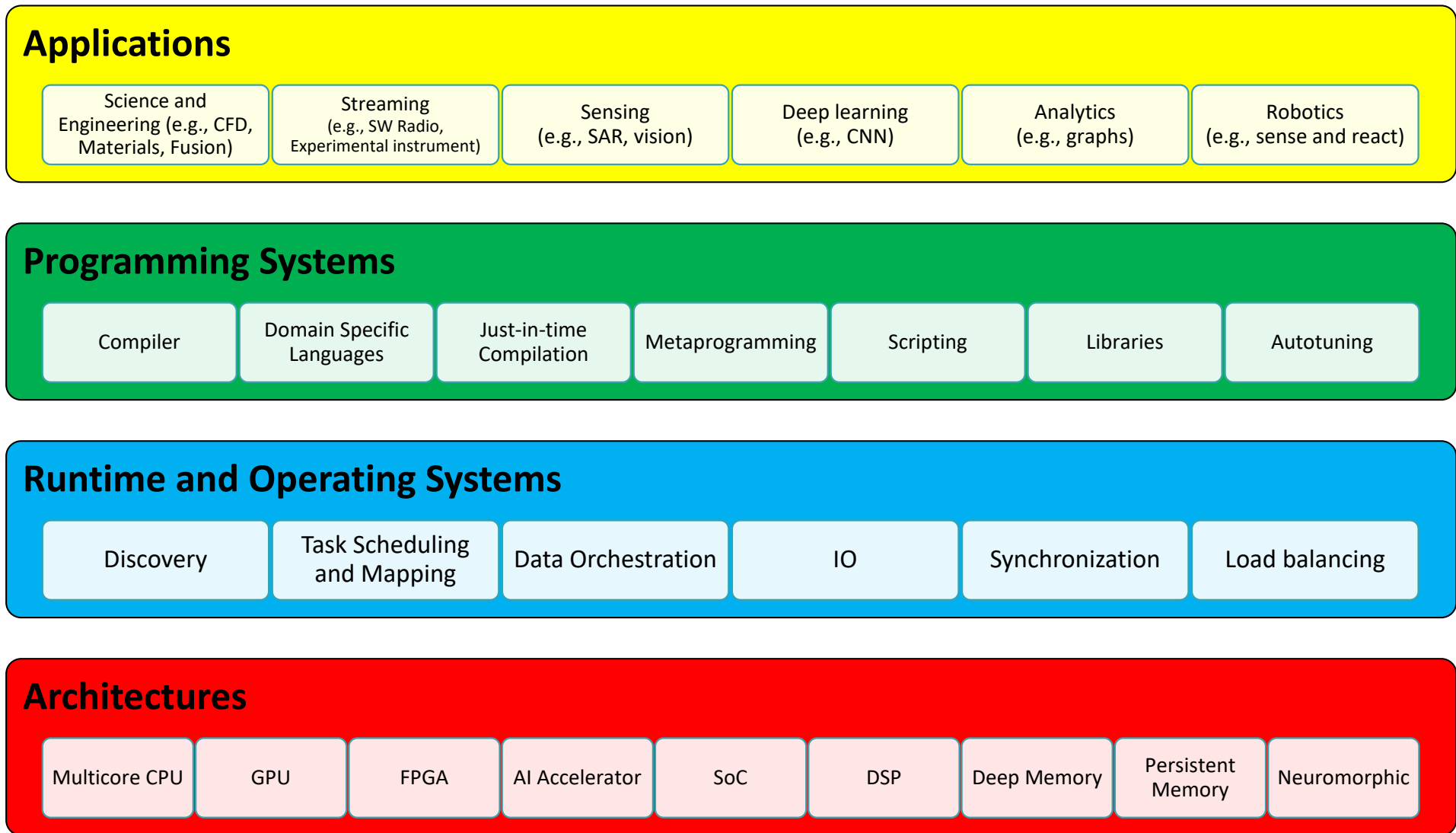
A Domain Specific Language (e.g., Claw, PySL)

A Domain Specific Framework (e.g., PetSc)

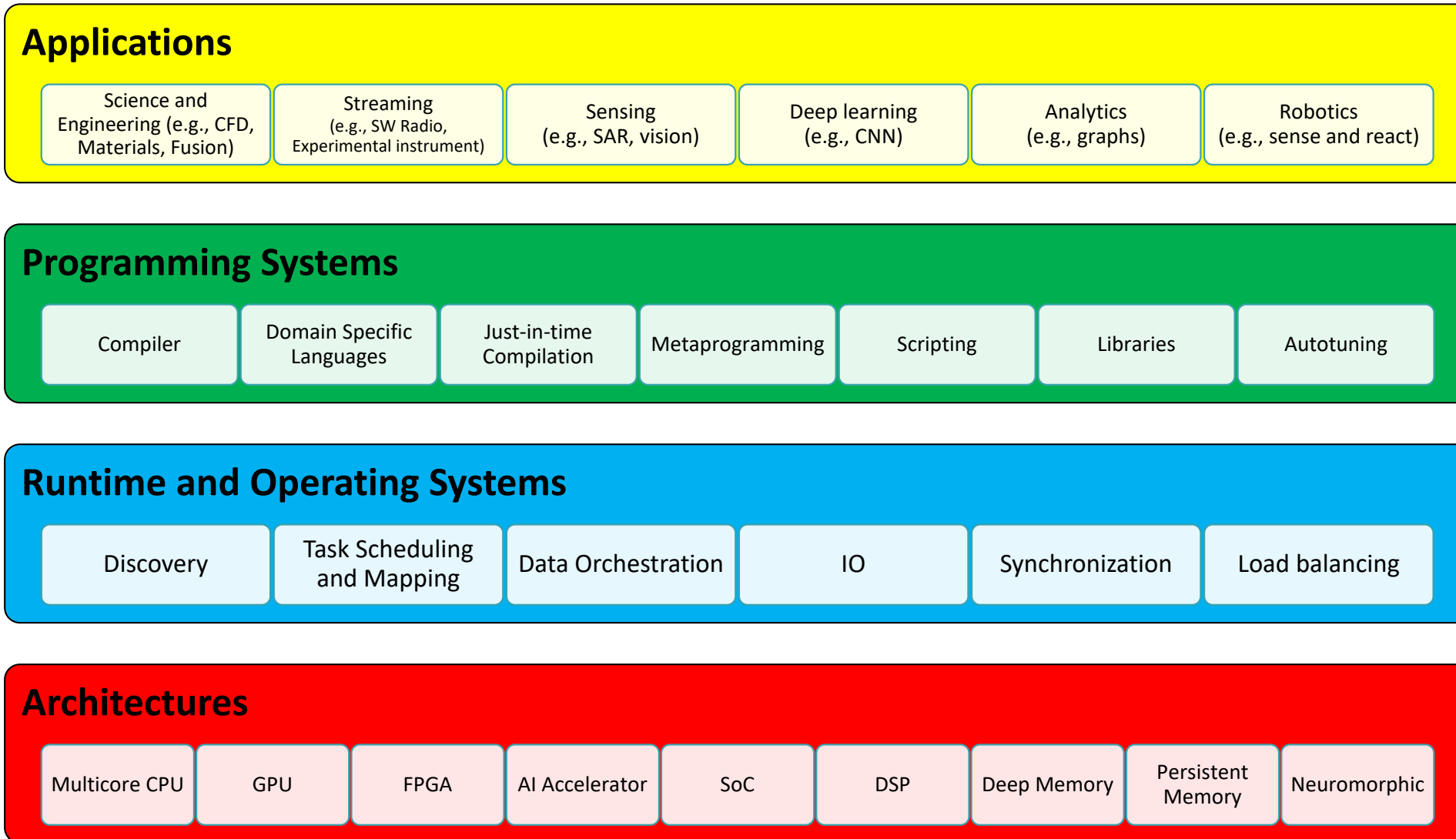
Some new unknown programming approach

All/some of the above

The FTG Vision



The FTG Vision | Applications

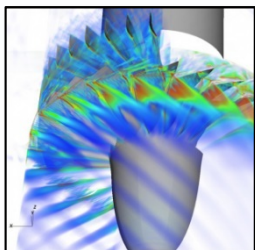
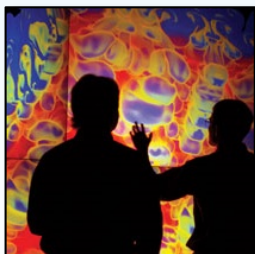


ECP applications target national problems in 6 strategic areas

National security

Stockpile stewardship

Next-generation electromagnetics simulation of hostile environment and virtual flight testing for hypersonic re-entry vehicles



Energy security

Turbine wind plant efficiency

High-efficiency, low-emission combustion engine and gas turbine design

Materials design for extreme environments of nuclear fission and fusion reactors

Design and commercialization of Small Modular Reactors

Subsurface use for carbon capture, petroleum extraction, waste disposal

Scale-up of clean fossil fuel combustion

Biofuel catalyst design

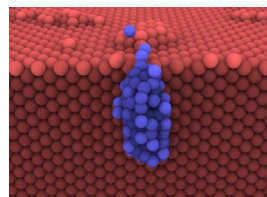
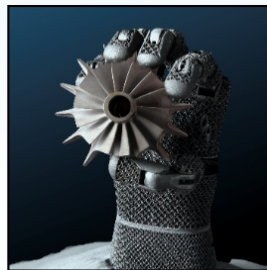
Economic security

Additive manufacturing of qualifiable metal parts

Reliable and efficient planning of the power grid

Seismic hazard risk assessment

Urban planning



Scientific discovery

Find, predict, and control materials and properties

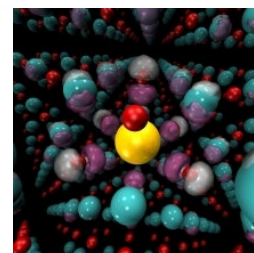
Cosmological probe of the standard model of particle physics

Validate fundamental laws of nature

Demystify origin of chemical elements

Light source-enabled analysis of protein and molecular structure and design

Whole-device model of magnetically confined fusion plasmas



Earth system

Accurate regional impact assessments in Earth system models

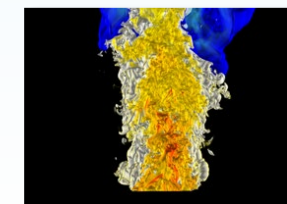
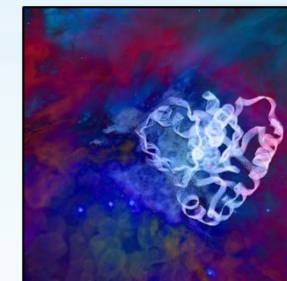
Stress-resistant crop analysis and catalytic conversion of biomass-derived alcohols

Metagenomics for analysis of biogeochemical cycles, climate change, environmental remediation



Health care

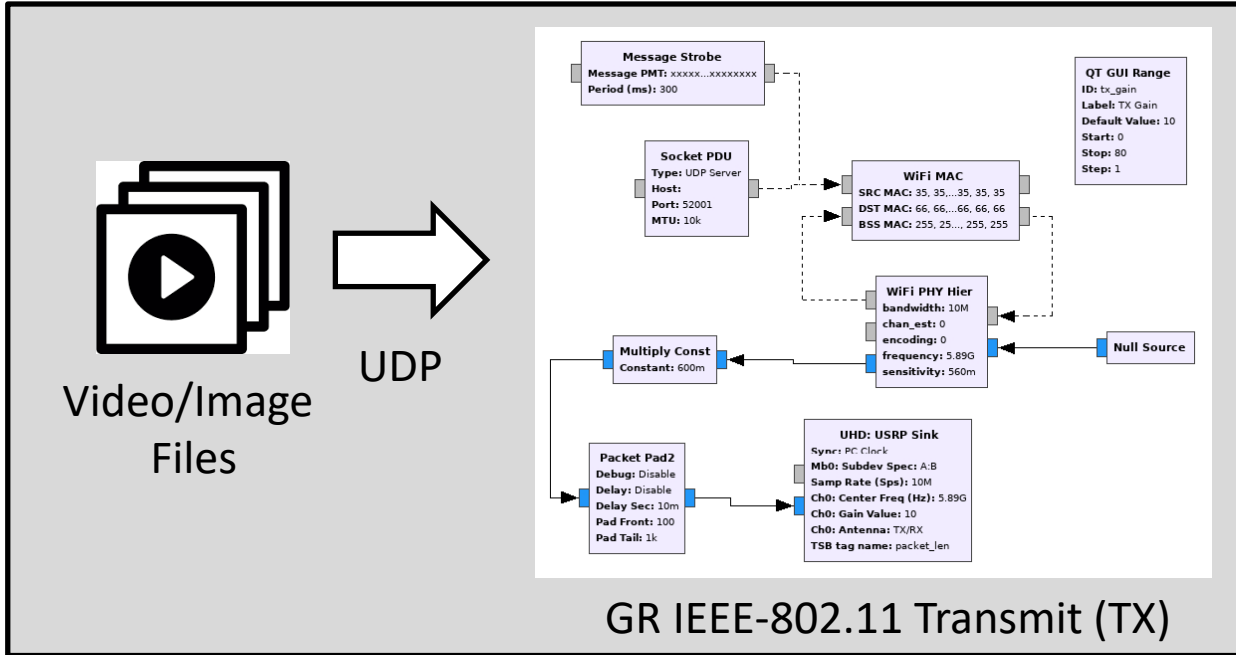
Accelerate and translate cancer research





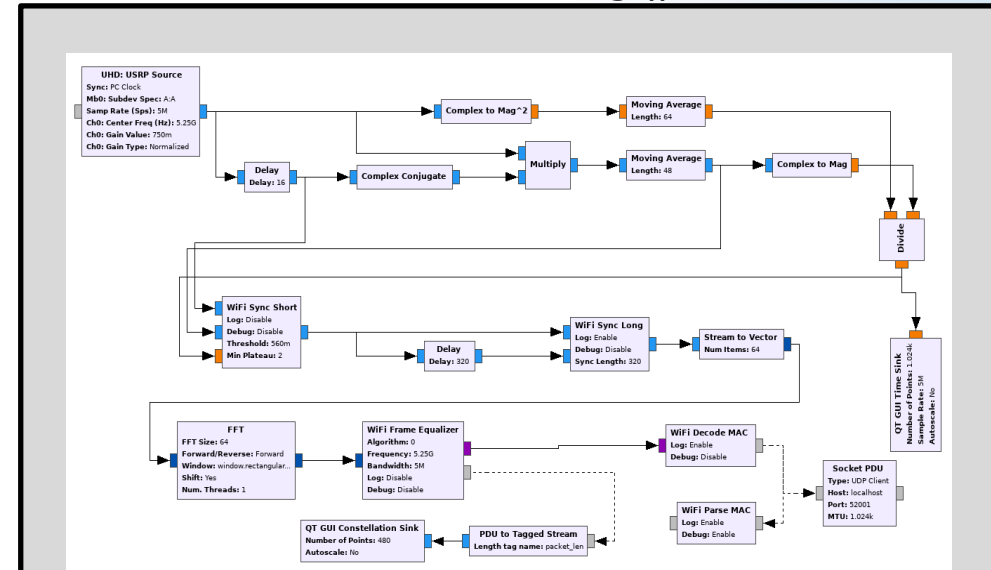
DARPA Domain Specific System on Chip Program is investigating Performance Portability of Software Defined Radio

Xavier SoC #1



GR IEEE-802.11 Transmit (TX)

Xavier SoC #2

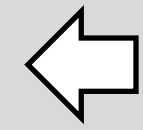


IEEE-802.11 Receive (RX)

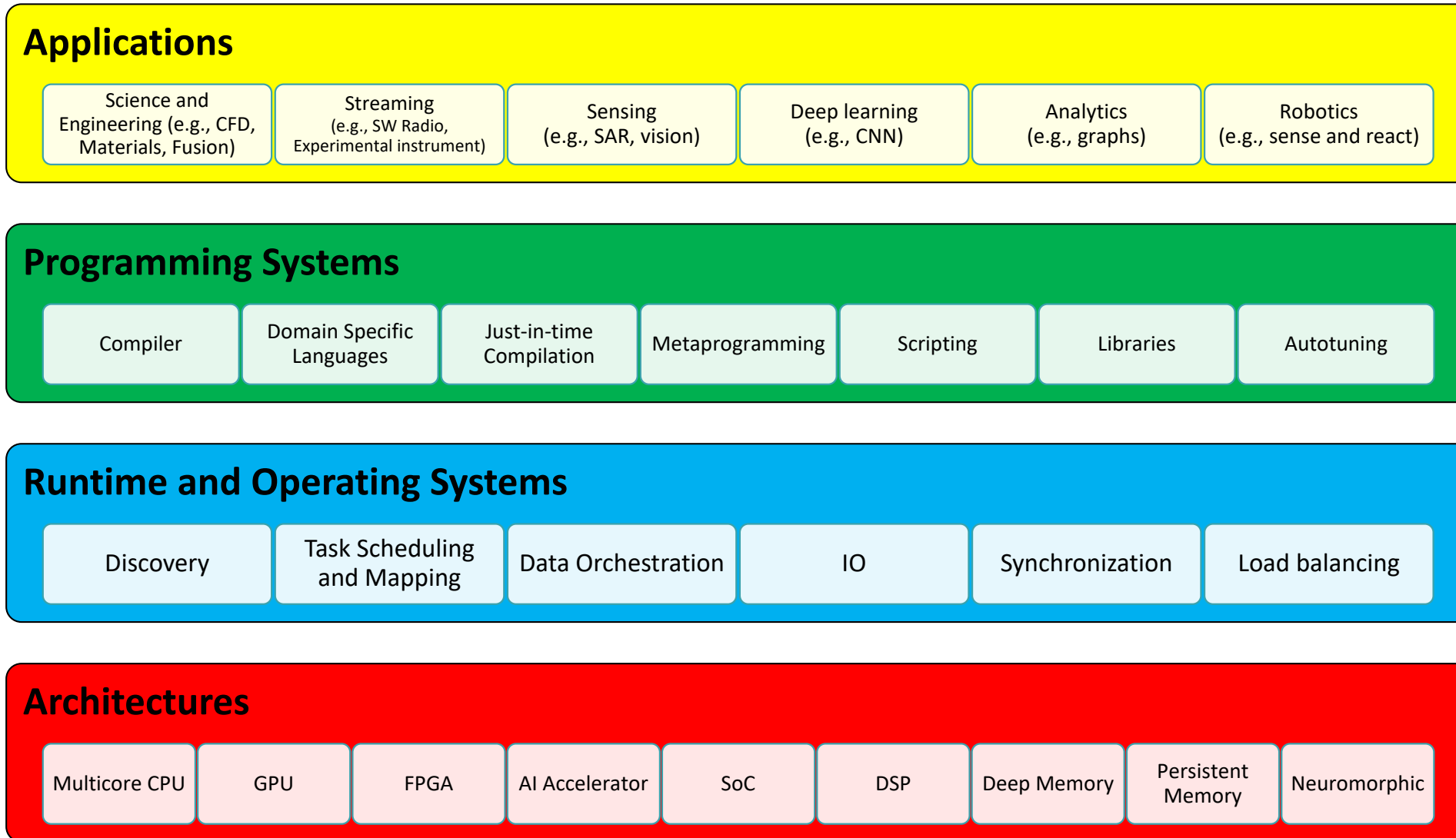
- Signal processing: An open-source implementation of IEEE-802.11 WIFI a/b/g with GR OOT modules.
- Input / Output file support via Socket PDU (UDP server) blocks
- Image/Video transcoding with OpenCL/OpenCV



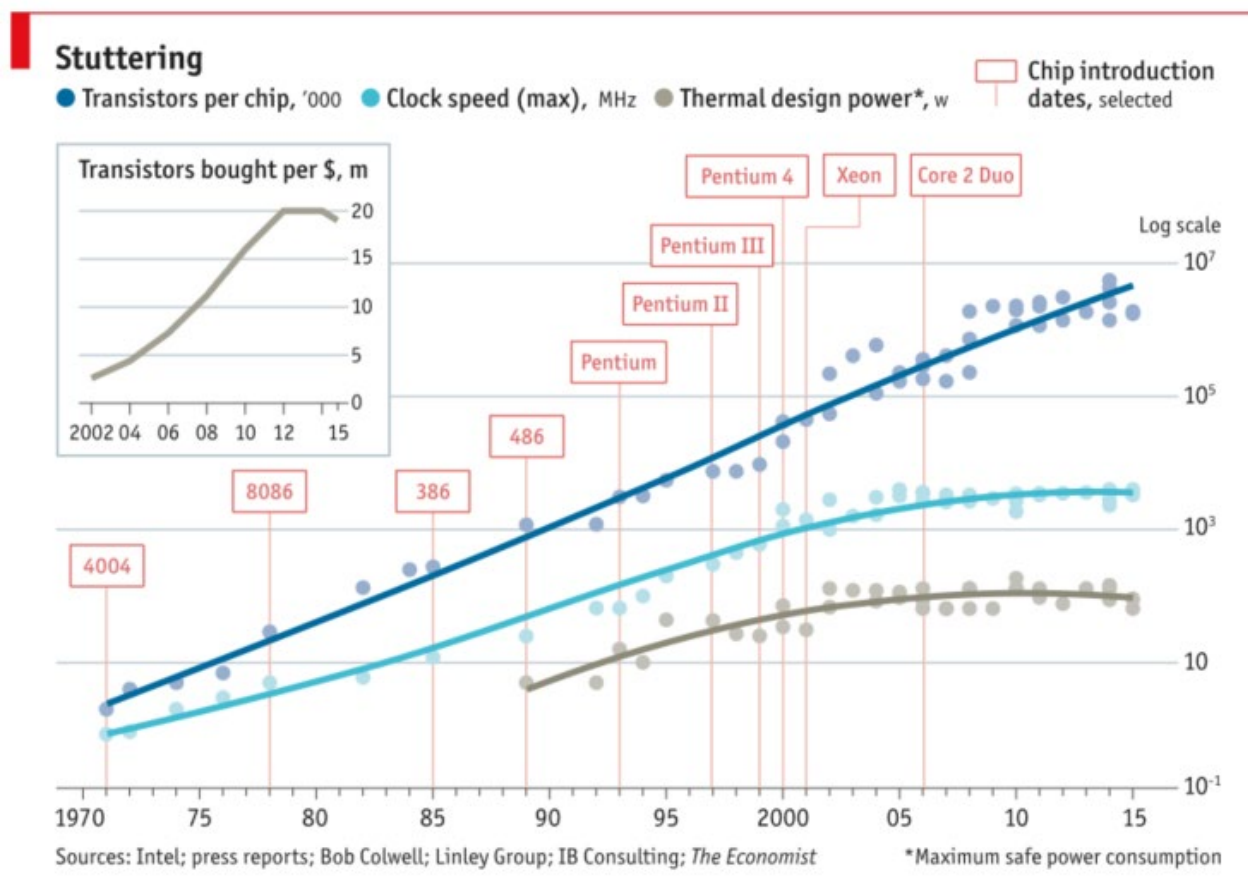
UDP



The FTG Vision | Architectures



Contemporary devices are approaching fundamental limits



Economist, Mar 2016

Dennard scaling has already ended. Dennard observed that voltage and current should be proportional to the linear dimensions of a transistor: 2x transistor count implies 40% faster and 50% more efficient.

R.H. Dennard, F.H. Gaensslen, V.L. Rideout, E. Bassous, and A.R. LeBlanc, "Design of ion-implanted MOSFET's with very small physical dimensions," *IEEE Journal of Solid-State Circuits*, 9(5):256-68, 1974,

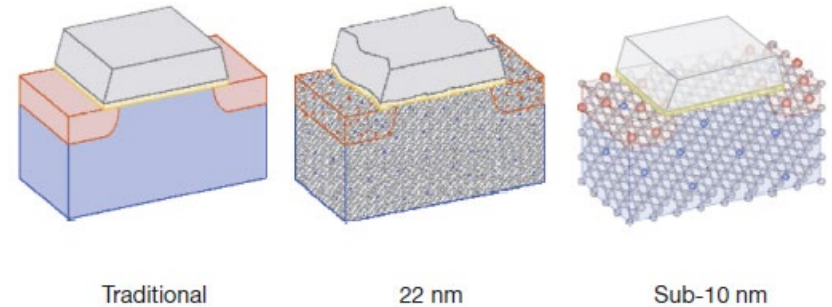


Figure 1 | As a metal oxide–semiconductor field effect transistor (MOSFET) shrinks, the gate dielectric (yellow) thickness approaches several atoms (0.5 nm at the 22-nm technology node). Atomic spacing limits the

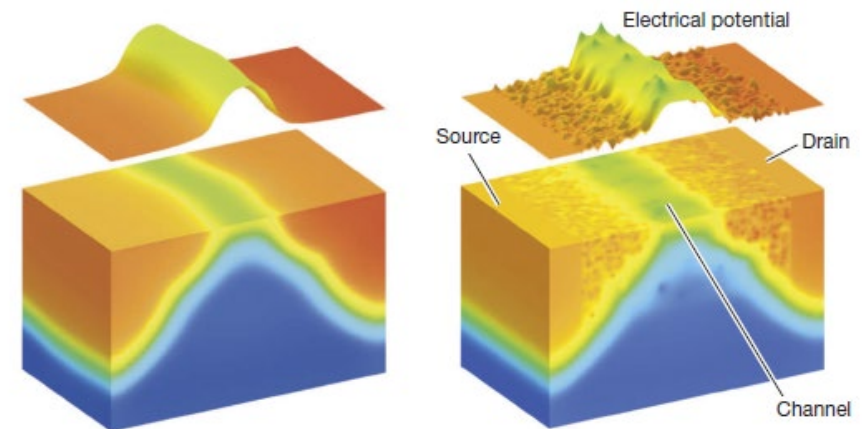
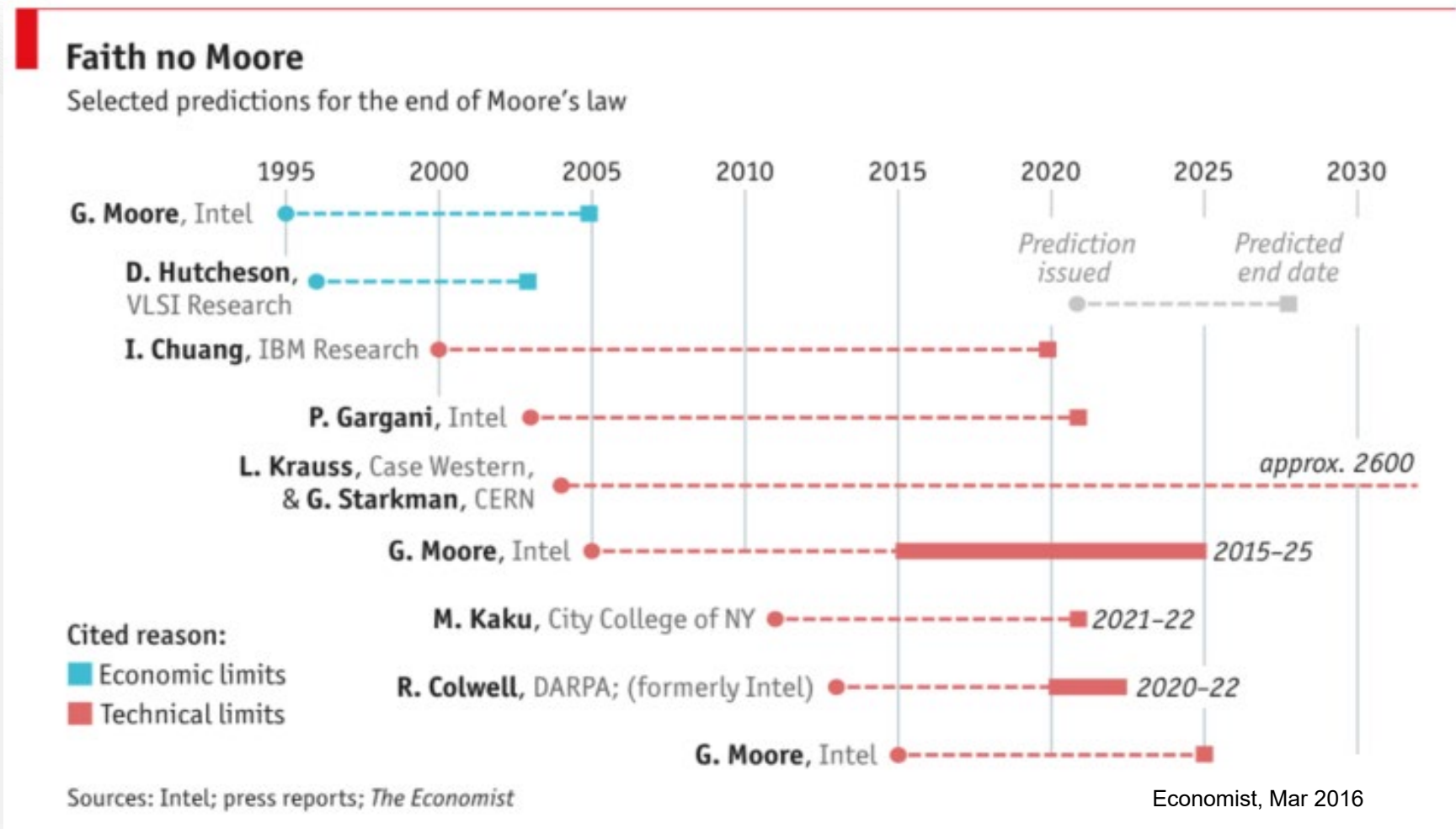


Figure 2 | As a MOSFET transistor shrinks, the shape of its electric field departs from basic rectilinear models, and the level curves become disconnected. Atomic-level manufacturing variations, especially for dopant

I.L. Markov, "Limits on fundamental limits to computation," *Nature*, 512(7513):147-54, 2014, doi:10.1038/nature13570.

End of Moore's Law : what's your prediction ??



News & Analysis

Foundries' Sales Show Hard Times Continuing

Peter Clarke

5/23/2016 09:33 PM EDT

2 comments

f Like 6
🐦 Tweet
in Share 43
G+

SEMICONDUCTOR ENGINEERING

and UMC, two of the industry's largest foundries, with recent sales that we expect to decline in the winter is not...

Uncertainty Grows For 5nm, 3nm

Nanosheets and nanowire FETs under development, but costs are skyrocketing. New packaging options could...



Samsung to Invest \$115 Billion in Foundry & Chip Businesses by 2030

GlobalFoundries Forfeit 7nm Manufacturing - EE Times Asia

Intel's 10nm Is Broken, Delayed Until 2019

37 COMMENTS

by Paul Alcorn April 26, 2018 at 6:30 PM

DESIGNLINES | WIRELESS AND NETWORKING DESIGNLINE

GlobalFoundries Selling ASIC Business to Marvell

By Dylan McGrath, 05.20.19

Share Post

Another Step Toward the End of Moore's Law

Samsung and TSMC move to 5-nanometer manufacturing

Number of Foundries with a Cutting Edge Logic Fab

SilTerra	X-FAB	Dongbu HiTek	ADI	Atmel	Rohm	Sanyo	Mitsubishi	ON	Hitachi	Cypress	Sony	Infineon	Sharp	Freescala	Renesas (NEC)	SMIC	Toshiba	Fujitsu	TI	Panasonic	STMicroelectronics	UMC	IBM	AMD	Samsung	TSMC	Intel	180 nm
			ADI	Atmel	Rohm	Sanyo	Mitsubishi	ON	Hitachi	Cypress	Sony	Infineon	Sharp	Freescala	Renesas	SMIC	Toshiba	Fujitsu	TI	Panasonic	STM	UMC	IBM	AMD	Samsung	TSMC	Intel	130 nm
																												90 nm
																												65 nm
																												45 nm/40 nm
																												32 nm/28 nm
																												22 nm/20 nm
																												16 nm/14 nm
																												10 nm
																												7 nm
																												5 nm

13 Dec 2019 | 20:20 GMT

TSMC's 5-Nanometer Process on Track for First Half of 2020

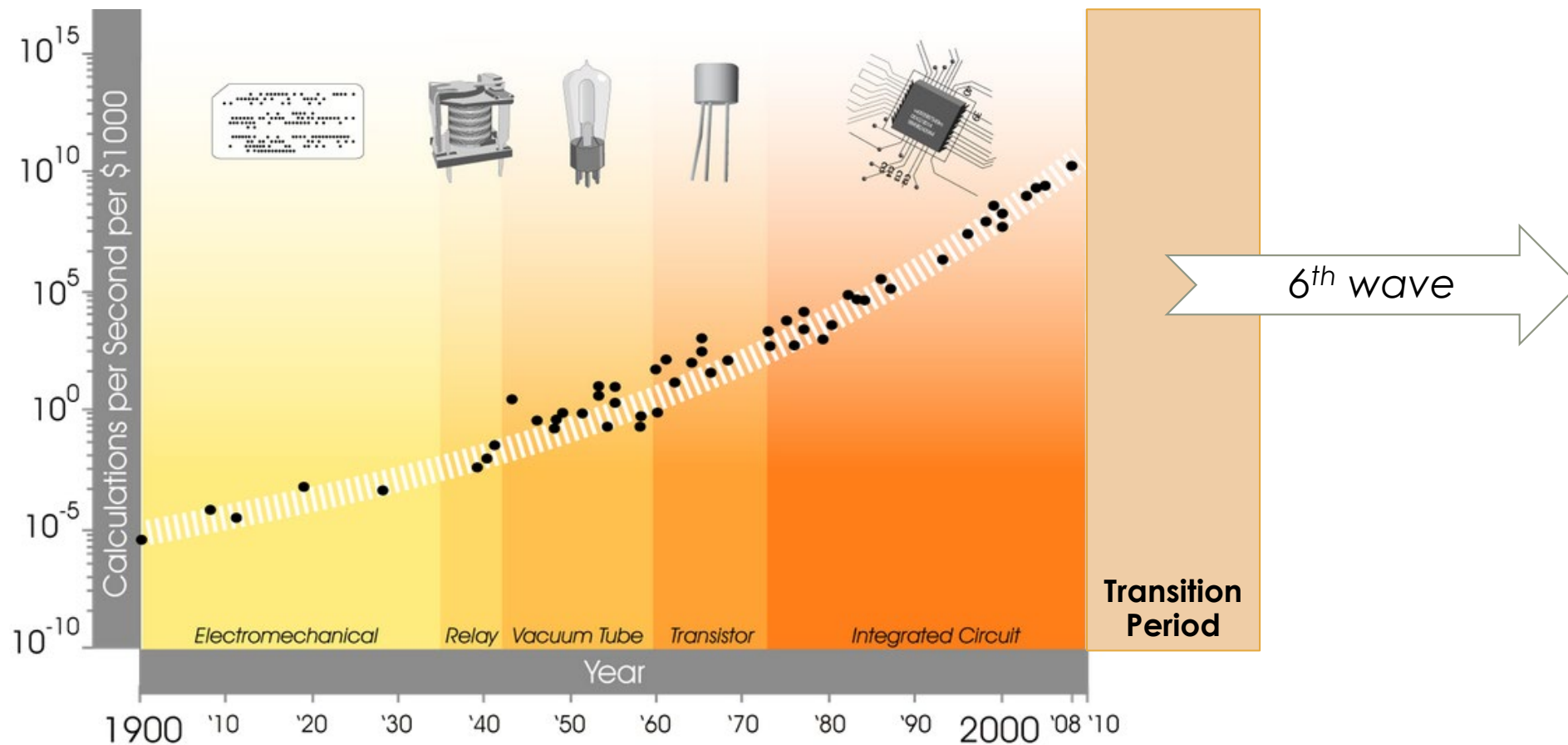
Devices are 15 percent faster, 30 percent more energy efficient

By Samuel K. Moore



Photo: Taiwan Semiconductor Manufacturing Co. The performance enhancement achieved by TSMC's new 5-nanometer process is partly due to the inclusion of a "high-mobility channel." How is it created? TSMC wouldn't reveal.

Sixth Wave of Computing



<http://www.kurzweilai.net/exponential-growth-of-computing>

Predictions for Transition Period

Optimize Software and Expose New Hierarchical Parallelism

- Redesign software to boost performance on upcoming architectures
- Exploit new levels of parallelism and efficient data movement

Architectural Specialization and Integration

- Use CMOS more effectively for specific workloads
- Integrate components to boost performance and eliminate inefficiencies
- Workload specific memory+storage system design

Emerging Technologies

- Investigate new computational paradigms
 - Quantum
 - Neuromorphic
 - Advanced Digital
 - Emerging Memory Devices

Predictions for Transition Period

Optimize Software and Expose New Hierarchical Parallelism

- Redesign software to boost performance on upcoming architectures
- Exploit new levels of parallelism and efficient data movement

Architectural Specialization and Integration

- Use CMOS more effectively for specific workloads
- Integrate components to boost performance and eliminate inefficiencies
- Workload specific memory+storage system design

Emerging Technologies

- Investigate new computational paradigms
 - Quantum
 - Neuromorphic
 - Advanced Digital
 - Emerging Memory Devices

Predictions for Transition Period

Optimize Software and Expose New Hierarchical Parallelism

- Redesign software to boost performance on upcoming architectures
- Exploit new levels of parallelism and efficient data movement

Architectural Specialization and Integration

- Use CMOS more effectively for specific workloads
- Integrate components to boost performance and eliminate inefficiencies
- Workload specific memory+storage system design

Emerging Technologies

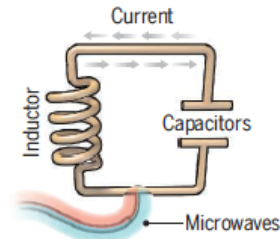
- Investigate new computational paradigms
 - Quantum
 - Neuromorphic
 - Advanced Digital
 - Emerging Memory Devices

Quantum computing: Qubit design and fabrication have made recent progress but still face challenges

Science 354, 1091 (2016) – 2 December

A bit of the action

In the race to build a quantum computer, companies are pursuing many types of quantum bits, or qubits, each with its own strengths and weaknesses.



Superconducting loops

A resistance-free current oscillates back and forth around a circuit loop. An injected microwave signal excites the current into superposition states.

Longevity (seconds)
0.00005

Logic success rate
99.4%

Number entangled
9

Company support

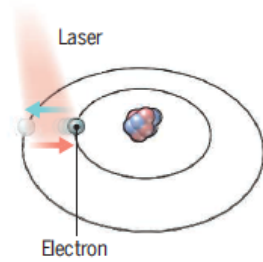
Google, IBM, Quantum Circuits

Pros

Fast working. Build on existing semiconductor industry.

Cons

Collapse easily and must be kept cold.



Trapped ions

Electrically charged atoms, or ions, have quantum energies that depend on the location of electrons. Tuned lasers cool and trap the ions, and put them in superposition states.

>1000

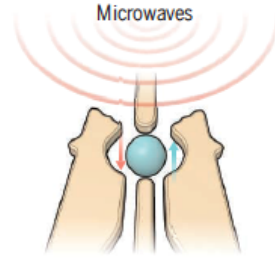
99.9%

14

ionQ

Very stable. Highest achieved gate fidelities.

Slow operation. Many lasers are needed.



Silicon quantum dots

These "artificial atoms" are made by adding an electron to a small piece of pure silicon. Microwaves control the electron's quantum state.

0.03

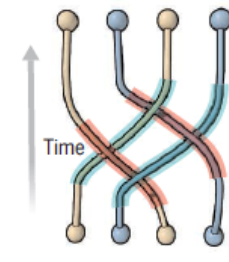
~99%

2

Intel

Stable. Build on existing semiconductor industry.

Only a few entangled. Must be kept cold.



Topological qubits

Quasiparticles can be seen in the behavior of electrons channeled through semiconductor structures. Their braided paths can encode quantum information.

N/A

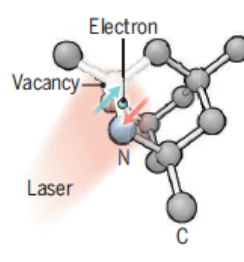
N/A

N/A

Microsoft, Bell Labs

Greatly reduce errors.

Existence not yet confirmed.



Diamond vacancies

A nitrogen atom and a vacancy add an electron to a diamond lattice. Its quantum spin state, along with those of nearby carbon nuclei, can be controlled with light.

10

99.2%

6

Quantum Diamond Technologies

Can operate at room temperature.

Difficult to entangle.

Note: Longevity is the record coherence time for a single qubit superposition state, logic success rate is the highest reported gate fidelity for logic operations on two qubits, and number entangled is the maximum number of qubits entangled and capable of performing two-qubit operations.

QUANTUM COMPUTING Progress and Prospects

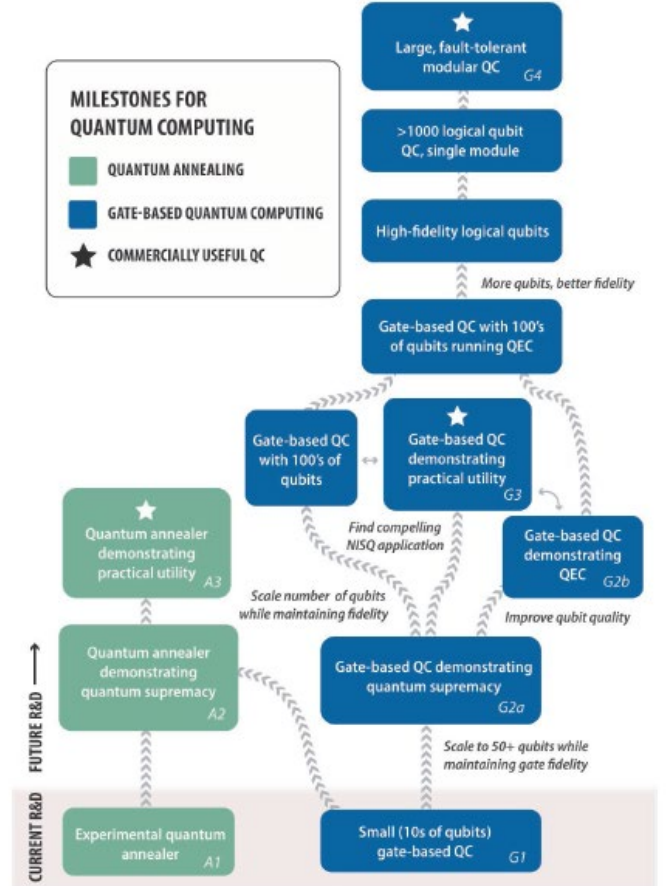


FIGURE 7.4 An illustration of potential milestones of progress in quantum computing. The arrangement of milestones corresponds to the order in which the committee thinks they are likely to be achieved; however, it is possible that some will not be achieved, or that they will not be achieved in the order indicated.

Fun Question: when was the field effect transistor patented?

Lilienfeld patents field effect transistor, October 8, 1926

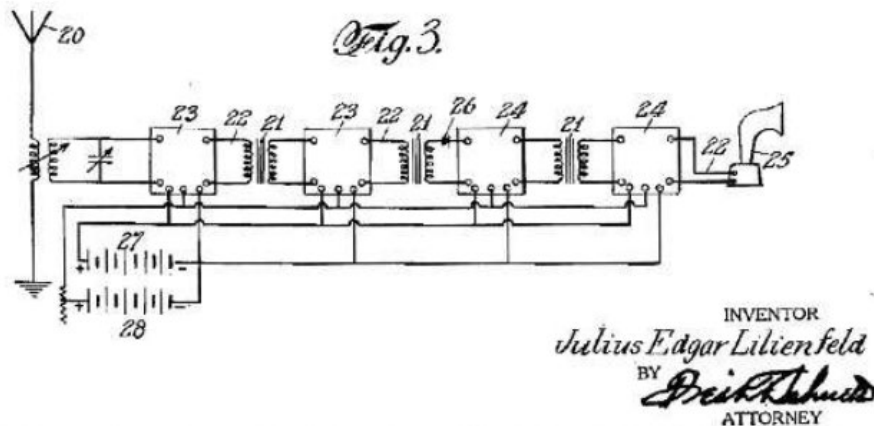
Jessica MacNeil -October 08, 2018

6 Comments

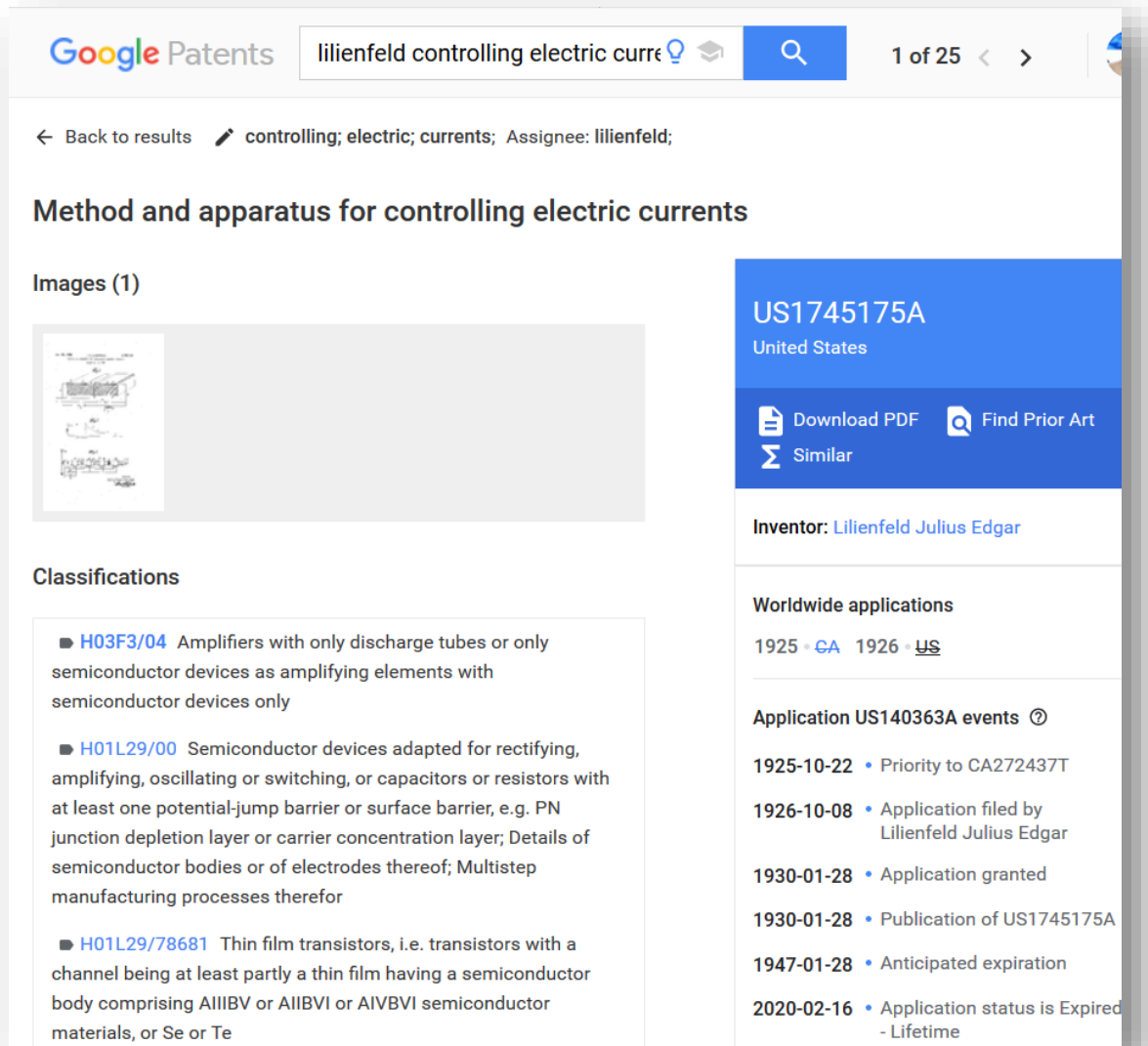
On this day in tech history, JE Lilienfeld filed a patent for a three-electrode structure using copper-sulfide semiconductor material, known today as a field-effect transistor.

Lilienfeld's patent for a "**method and apparatus for controlling electric currents**" was granted on January 28, 1930.

According to the patent, his invention was for controlling the flow of electric current between two terminals of an electrically conducting solid by establishing a third potential between the terminals, particularly for the amplification of oscillating currents like those in radio communication.



<https://www.edn.com/electronics-blogs/edn-moments/4422371/Lilienfeld-patents-field-effect-transistor--October-8--1926>




Google Patents lilienfeld controlling electric current 1 of 25

← Back to results ✎ controlling; electric; currents; Assignee: lilienfeld;

Method and apparatus for controlling electric currents

Images (1)



US1745175A
United States

Download PDF Find Prior Art
Similar

Inventor: Lilienfeld Julius Edgar

Classifications

- **H03F3/04** Amplifiers with only discharge tubes or only semiconductor devices as amplifying elements with semiconductor devices only
- **H01L29/00** Semiconductor devices adapted for rectifying, amplifying, oscillating or switching, or capacitors or resistors with at least one potential-jump barrier or surface barrier, e.g. PN junction depletion layer or carrier concentration layer; Details of semiconductor bodies or of electrodes thereof; Multistep manufacturing processes therefor
- **H01L29/78681** Thin film transistors, i.e. transistors with a channel being at least partly a thin film having a semiconductor body comprising AIIIBV or AIIIVI or AIVBVI semiconductor materials, or Se or Te

Worldwide applications

1925 · CA 1926 · US

Application US140363A events

- 1925-10-22 · Priority to CA272437T
- 1926-10-08 · Application filed by Lilienfeld Julius Edgar
- 1930-01-28 · Application granted
- 1930-01-28 · Publication of US1745175A
- 1947-01-28 · Anticipated expiration
- 2020-02-16 · Application status is Expired - Lifetime

Predictions for Transition Period

Optimize Software and Expose New Hierarchical Parallelism

- Redesign software to boost performance on upcoming architectures
- Exploit new levels of parallelism and efficient data movement

Architectural Specialization and Integration

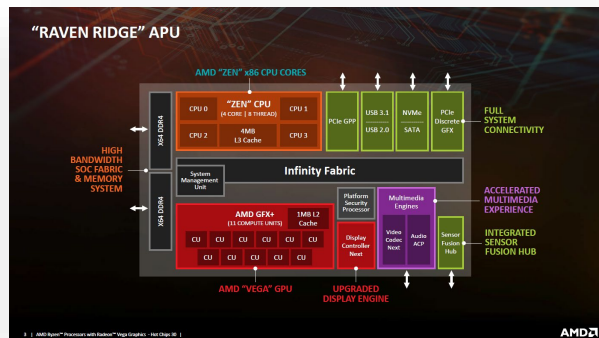
- Use CMOS more effectively for specific workloads
- Integrate components to boost performance and eliminate inefficiencies
- Workload specific memory+storage system design

Emerging Technologies

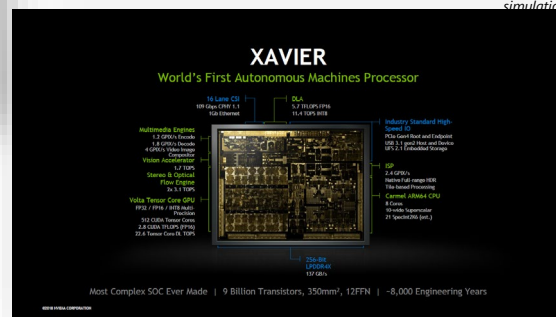
- Investigate new computational paradigms
 - Quantum
 - Neuromorphic
 - Advanced Digital
 - Emerging Memory Devices

Pace of Architectural Specialization is Quickening

- Industry, lacking Moore's Law, will need to continue to differentiate products (to stay in business)
 - Use the same transistors differently to enhance performance
- Architectural design will become extremely important, critical
 - Dark Silicon
 - Address new parameters for benefits/curse of Moore's Law
- 50+ new companies focusing on hardware for Machine Learning



HotChips 2018



HotChips 2018

Intel's Nervana AI platform takes aim at Nvidia's GPU technology

Firm claims Xeon-based chips will deliver a '100-fold increase' in deep learning performance



CHIPMAKER Intel has set out its plans for artificial intelligence (AI) and claimed that it will reduce the time to train a deep learning model by up to 100 times within the next three years.

At the forefront of the firm's AI ambitions is the Intel Nervana platform, which was announced on Thursday following Intel's acquisition of deep learning startup Nervana Systems earlier this year.

<http://www.theinquirer.net/inquirer/news/2477796/intels-nervana-ai-platform-takes-aim-at-nvidias-gpu-technology>



D.E. Shaw, M.M. Deneroff, R.O. Dror et al., "Anton, a special-purpose machine for molecular dynamics simulation," *Communications of the ACM*, 51(7):91-7, 2008.

GOOGLE BUILT ITS VERY OWN CHIPS TO POWER ITS AI BOTS



GOOGLE

GOOGLE HAS DESIGNED its own computer chip for driving deep neural networks, an AI technology that is reinventing the way Internet services operate.

This morning at Google I/O, the centerpiece of the company's year, CEO Sundar Pichai said that Google has designed an ASIC, or application-specific integrated circuit, that's specific to deep neural nets. These are networks of

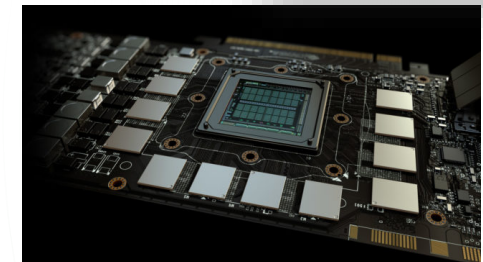
<http://www.wired.com/2016/05/google-tpu-custom-chips/>

NEW AT AMAZON: ITS OWN CHIPS FOR CLOUD COMPUTING

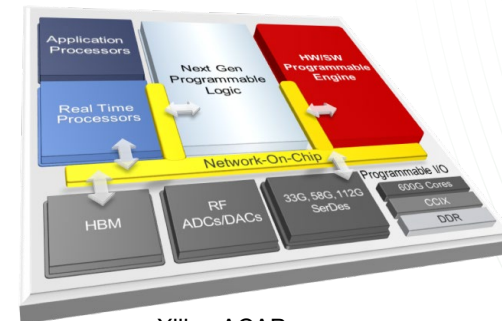


Amazon Web Services CEO Andy Jassy speaks at an event in San Francisco in 2017. DAVID PAUL MORRIS/BLOOMBERG/GETTY IMAGES

BIG SOFTWARE COMPANIES don't just stick to software any more—they build computer chips. The latest proof comes from Amazon, which announced late Monday that its cloud computing division has created its own chips to power customers' websites and other services. The chips, dubbed Graviton, are built around the same technology that powers smartphones and tablets. That approach has been much discussed in the cloud industry but never



<https://fossbytes.com/nvidia-volta-gddr6-2018/>



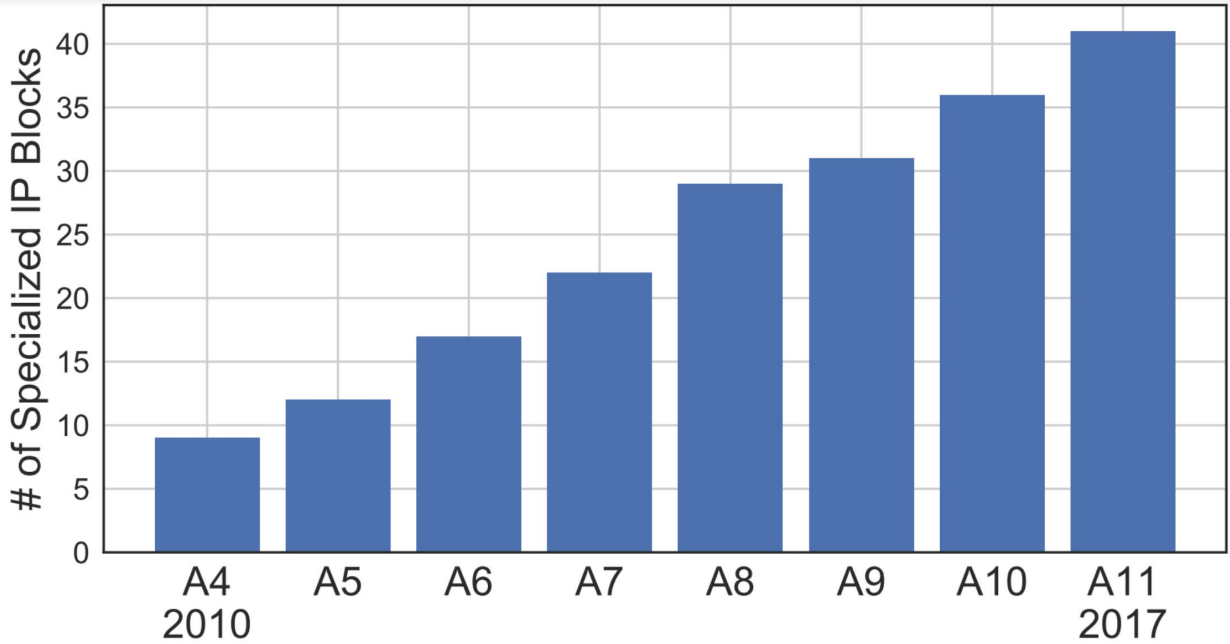
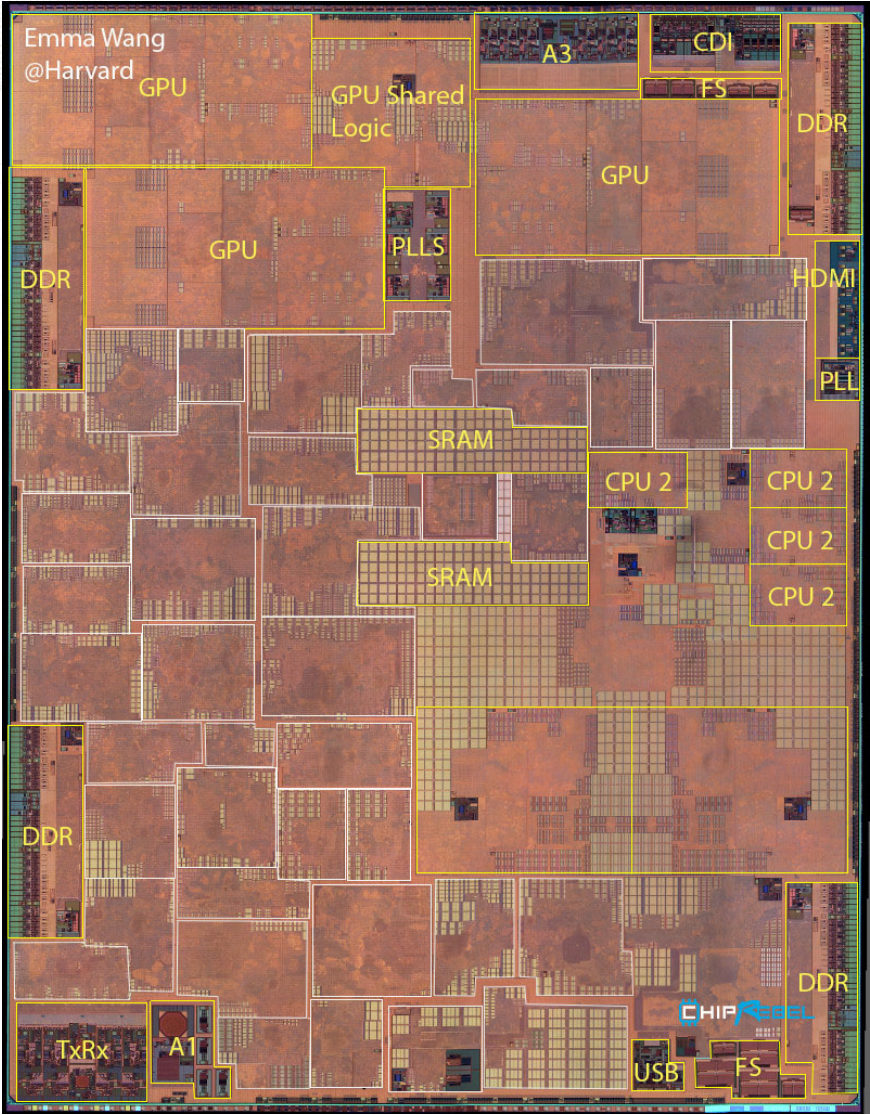
Xilinx ACAP



<https://www.thebroadcastbridge.com/content/entry/1094/altera-announces-national-laboratory>



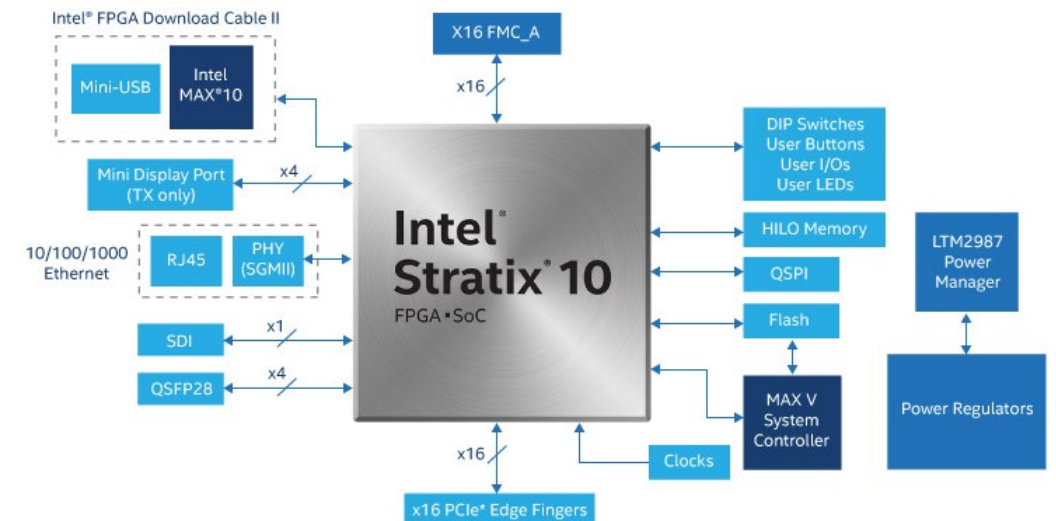
Analysis of Apple A-* SoCs



Intel Stratix 10 FPGA

Experimental Computing Lab (ExCL) managed by the ORNL Future Technologies Group

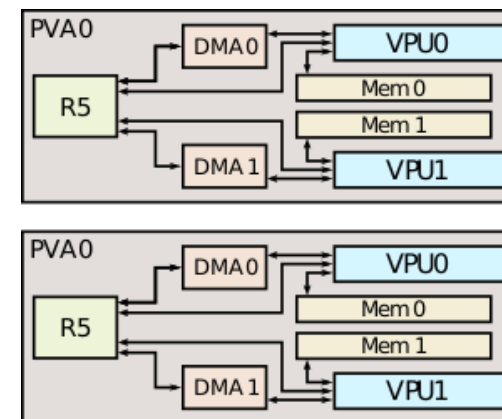
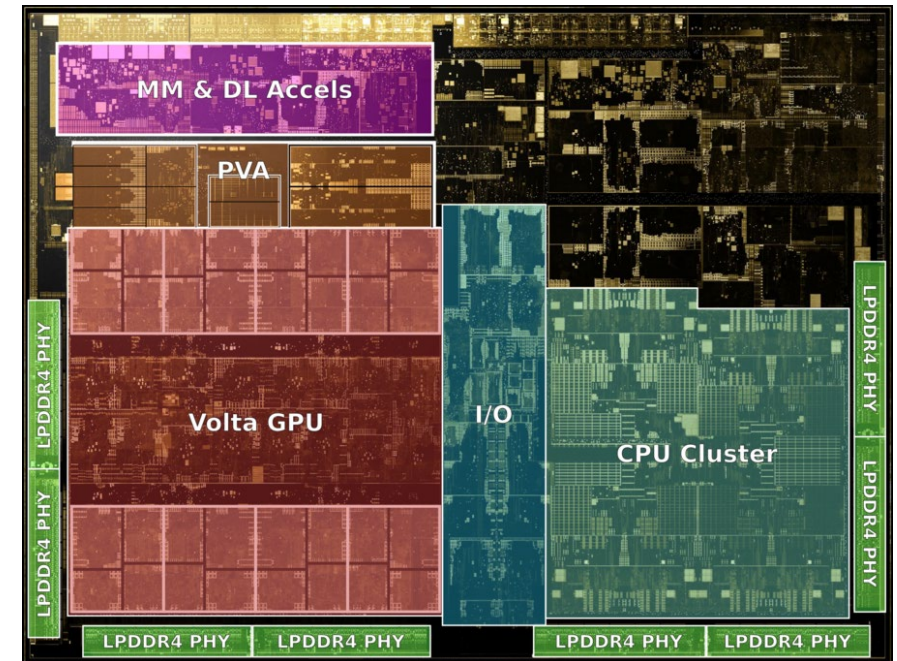
- Intel Stratix 10 FPGA and four banks of DDR4 external memory
 - Board configuration: Nallatech 520 Network Acceleration Card
- Up to 10 TFLOPS of peak single precision performance
- 25MBytes of L1 cache @ up to 94 TBytes/s peak bandwidth
- 2X Core performance gains over Arria® 10
- Quartus and OpenCL software (Intel SDK v18.1) for using FPGA
- Provide researcher access to advanced FPGA/SOC environment



NVIDIA Jetson AGX Xavier SoC

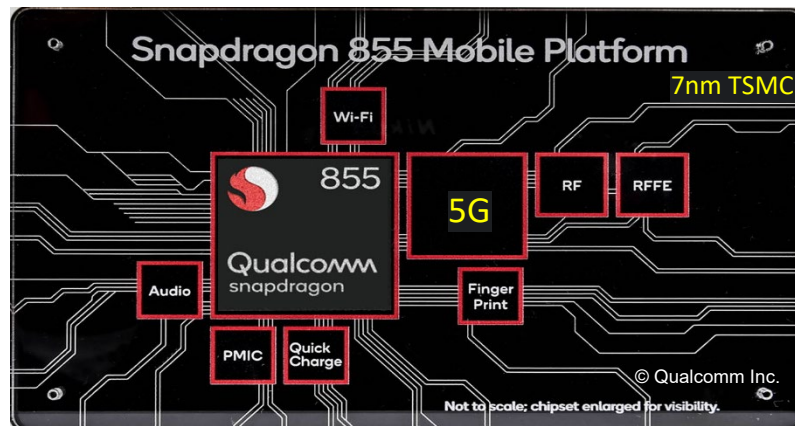
Experimental Computing Lab (ExCL) managed by the ORNL Future Technologies Group

- NVIDIA Jetson AGX Xavier:
- High-performance system on a chip for autonomous machines
- Heterogeneous SoC contains:
 - Eight-core 64-bit ARMv8.2 CPU cluster (Carmel)
 - 1.4 CUDA TFLOPS (FP32) GPU with additional inference optimizations (Volta)
 - 11.4 DL TOPS (INT8) Deep learning accelerator (NVDLA)
 - 1.7 CV TOPS (INT8) 7-slot VLIW dual-processor Vision accelerator (PVA)
 - A set of multimedia accelerators (stereo, LDC, optical flow)
- Provides researchers access to advanced high-performance SOC environment

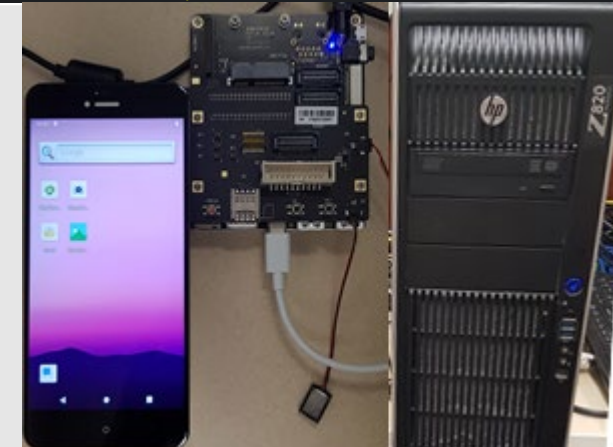


Qualcomm 855 SoC (SM8510P) Snapdragon™

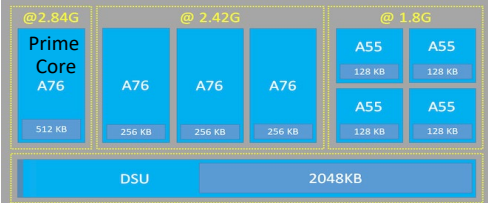
Experimental Computing Lab (ExCL) managed by the ORNL Future Technologies Group



Qualcomm Development Board connected to (mcmurdo) HPZ820



Kyro 485 (8-ARM Prime+BigLittle Cores)



Hexagon 690 (DSP + AI)

- Quad threaded Scalar Core
- DSP + 4 Hexagon Vector Xccelerators
- New Tensor Xccelerator for AI
- Apps: AI, Voice Assistance, AV codecs

Adreno 640

- Vulkan, OpenCL, OpenGL ES 3.1
- Apps: HDR10+, HEVC, Dolby, etc
- Enables 8k-360° VR video playback
- 20% faster compared to Adreno 630

Connectivity (5G)

- Snapdragon X24 LTE (855 built-in) modem LTE Category 20
- Snapdragon X50 5G (external) modem (for 5G devices)
- Qualcomm Wi-Fi 6-ready mobile platform: (802.11ax-ready, 802.11ac Wave 2, 802.11ay, 802.11ad)
- Qualcomm 60 GHz Wi-Fi mobile platform: (802.11ay, 802.11ad)
- Bluetooth Version: 5.0
- Bluetooth Speed: 2 Mbps
- High accuracy location with dual-frequency GNSS.

Spectra 360 ISP

- New dedicated Image Signal Processor (ISP)
- Dual 14-bit CV-ISPs; 48MP @ 30fps single camera
- Hardware CV for object detection, tracking, stereo depth process
- 6DoF XR Body tracking, H265, 4K60 HDR video capture, etc.

- Connected Qualcomm board to HPZ820 through USB
- Development Environment: Android SDK/NDK
- Login to mcmurdo machine
 - \$ ssh -Y mcmurdo
- Setup Android platform tools and development environment
 - \$ source /home/nqx/setup_android.source
- Run Hello-world on ARM cores
 - \$ git clone <https://code.ornl.gov/nqx/helloworld-android>
 - \$ make compile push run
- Run OpenCL example on GPU
 - \$ git clone <https://code.ornl.gov/nqx/opencl-img-processing>
 - Run Sobel edge detection
 - \$ make compile push run fetch
- Login to Qualcomm development board shell
 - \$ adb shell
 - \$ cd /data/local/tmp

Growing Open Source Hardware Movement Enables Rapid Chip Design



RISC-V Ecosystem

Software

Open-source software:

Gcc, binutils, glibc, Linux, BSD, LLVM, QEMU, FreeRTOS, ZephyrOS, LiteOS, SylixOS, ...

Commercial software:

Lauterbach, Segger, Micrium, ExpressLogic, ...



ISA specification

Golden Model

Compliance

Hardware

Open-source cores:

Rocket, BOOM, RI5CY, Ariane, PicoRV32, Piccolo, SCR1, Hummingbird, ...

Commercial core providers:

Andes, Bluespec, Cloudbear, Codaip, Cortus, C-Sky, Nuclei, SiFive, Syntacore, ...

Inhouse cores:

Nvidia, +others

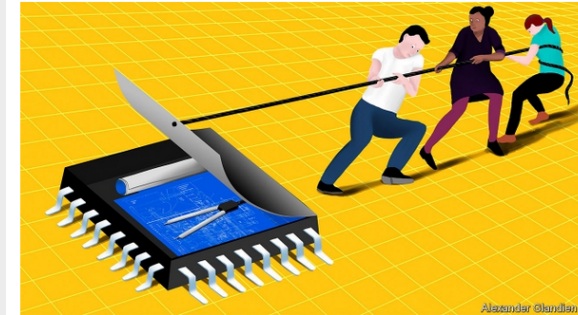
7

RISC-V Summit, 2018

Open-source computing

A new blueprint for microprocessors challenges the industry's giants

RISC-V is an alternative to proprietary designs



Print edition | Science and technology >
Oct 3rd 2019



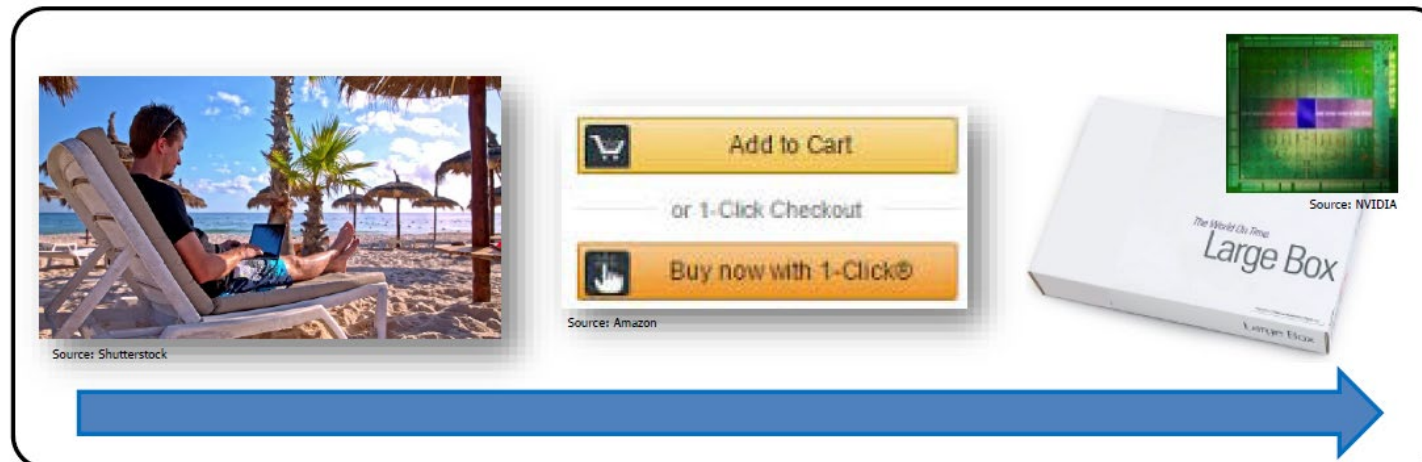
MOST MICROPROCESSORS—the chips that do the grunt work in computers—are built around designs, known as instruction-set architectures (ISAs), which are owned either by Intel, an American giant, or by Arm, a Japanese one. Intel's ISAs power desktop computers, servers and laptops. Arm's power phones, watches and other mobile devices. Together, these two firms dominate the market. Almost every one of the 5.1bn mobile phones on the planet, for example, relies on an Arm-designed ISA. The past year, however, has seen a boomlet in chips made using an ISA called RISC-V. If boomlet becomes boom, it may change the chip industry dramatically, to the detriment of Arm and Intel, because unlike the ISAs from those two firms, which are proprietary, RISC-V is available to anyone, anywhere, and is free.

An ISA is a standardised description of how a chip works at the most basic level, and instructions for writing software to run on it. To draw an analogy, a house might have two floors or three, five bedrooms or six, one bathroom or two. That is up to the architect. An ISA, however, is the equivalent of insisting that the same sorts of electrical sockets and water inlets and outlets be put in the same places in every appropriate room, so that an electrician or a plumber can find them instantly and carry the correct kit to connect to them.

DARPA ERI Programs Aiming for Agile (and Frequent) Chip Creation

IDEA/POSH End State – A Universal Hardware Compiler

```
$ git clone https://github.com/darpa/idea  
$ git clone https://github.com/darpa/posh  
$ cd posh  
$ make soc42
```



Distribution Statement "A" (Approved for Public Release, Distribution Unlimited)

23

A. Olofsson, 2018

Summary:

Transition Period will be Disruptive – Opportunities and Pitfalls Abound

- New devices and architectures may not be hidden in traditional levels of abstraction
- Examples
 - A new type of CNT transistor may be completely hidden from higher levels
 - A new paradigm like quantum may require new architectures, programming models, and algorithmic approaches

Layer	Switch, 3D	NVM	Approximate	Neuro	Quantum
<i>Application</i>	1	1	2	2	3
<i>Algorithm</i>	1	1	2	3	3
<i>Language</i>	1	2	2	3	3
<i>API</i>	1	2	2	3	3
<i>Arch</i>	1	2	2	3	3
<i>ISA</i>	1	2	2	3	3
<i>Microarch</i>	2	3	2	3	3
<i>FU</i>	2	3	2	3	3
<i>Logic</i>	3	3	2	3	3
<i>Device</i>	3	3	2	3	3

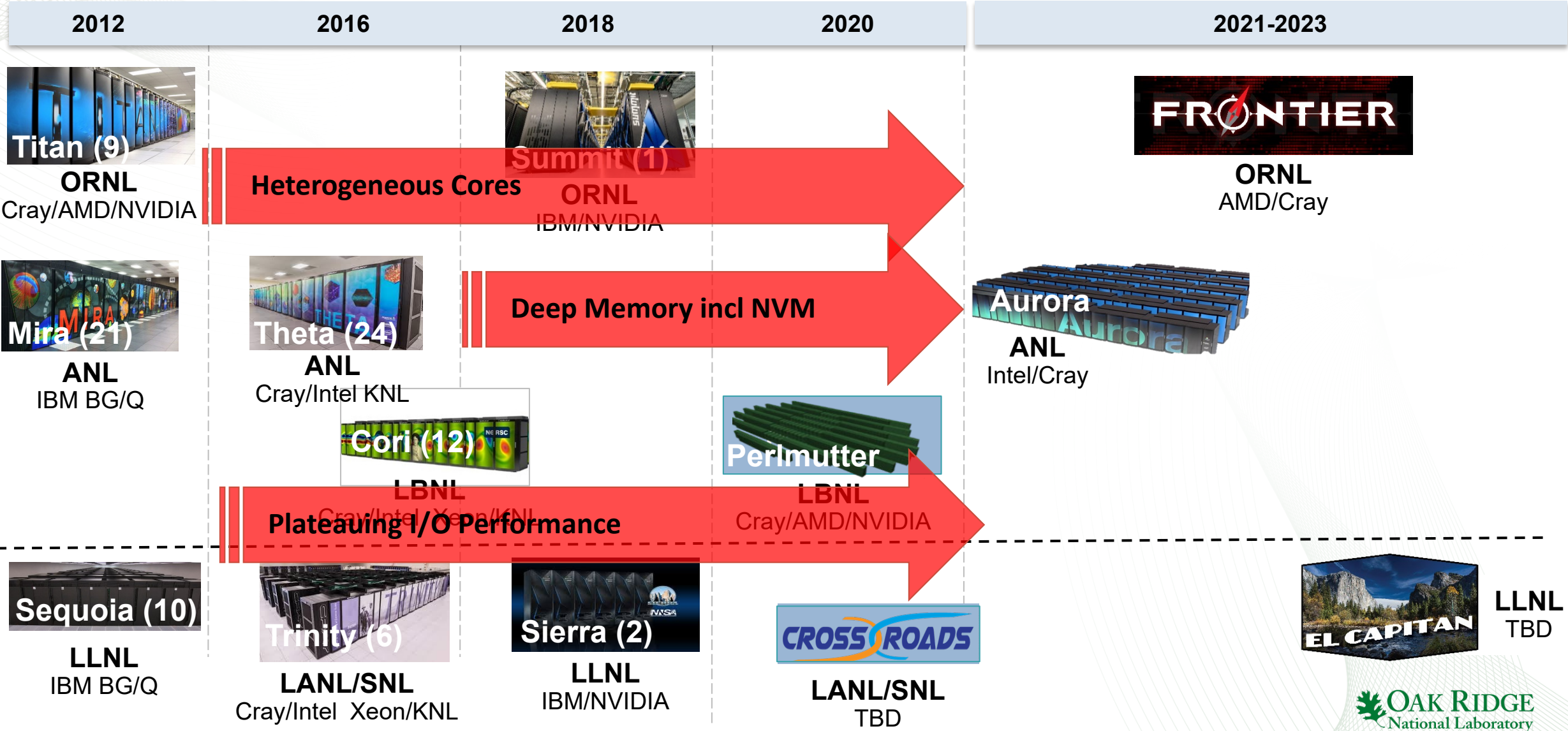
Adapted from IEEE Rebooting Computing Chart

Department of Energy (DOE) Roadmap to Exascale Systems

An impressive, productive lineup of *accelerated node* systems supporting DOE's mission

Pre-Exascale Systems [Aggregate Linpack (Rmax) = 323 PF!]

First U.S. Exascale Systems

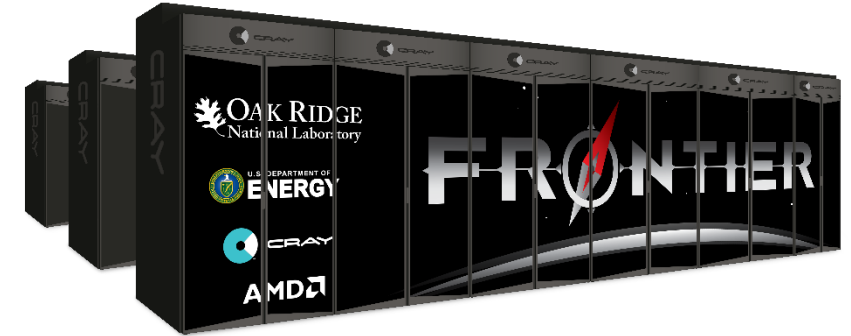


Jan 2018



Frontier Continues the Accelerated Node Design

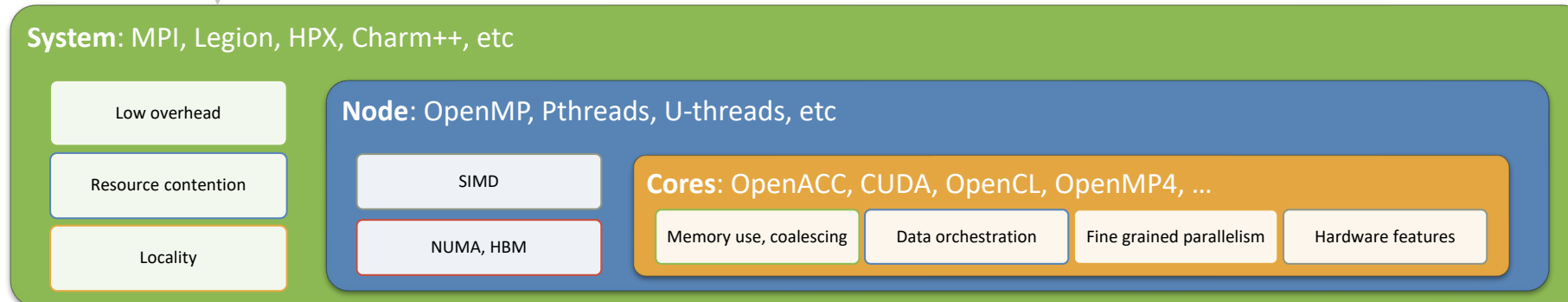
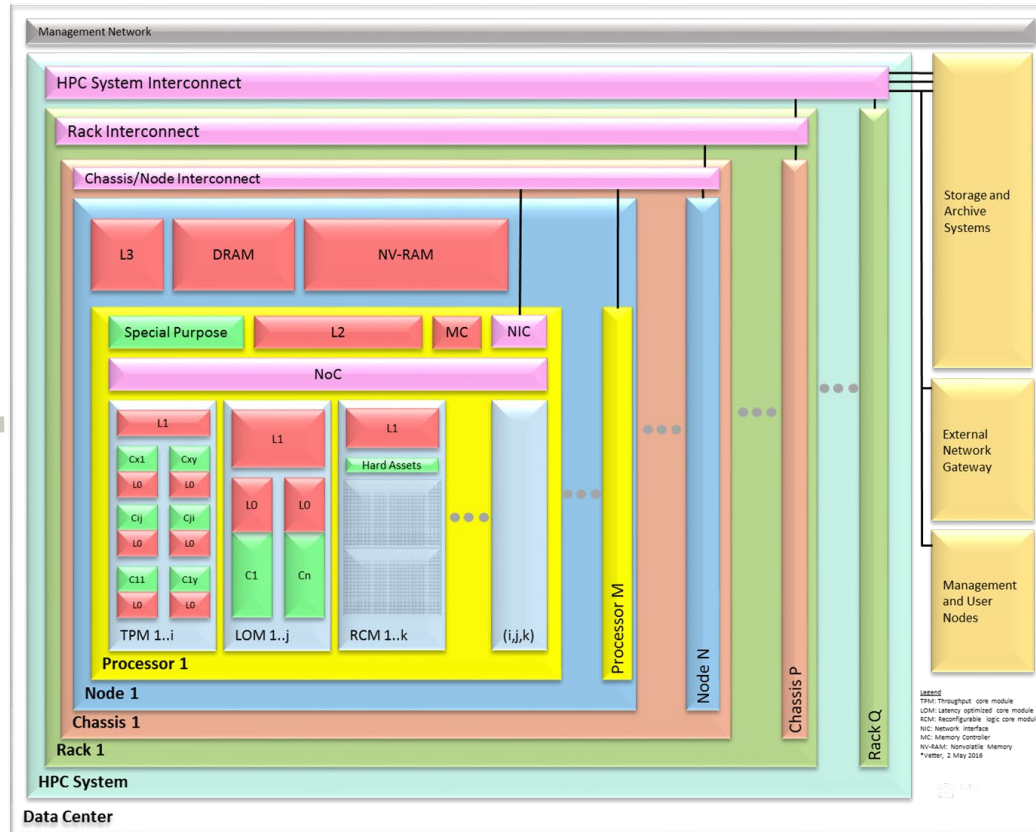
- Partnership between ORNL, Cray, and AMD
- The Frontier system will be delivered in 2021
- Peak Performance greater than 1.5 EF
- Composed of more than 100 Cray Shasta cabinets
 - Connected by Slingshot™ interconnect with adaptive routing, congestion control, and quality of service
- Accelerated Node Architecture:
 - One purpose-built AMD EPYC™ processor
 - Four HPC and AI optimized Radeon Instinct™ GPU accelerators
 - Fully connected with high speed AMD Infinity Fabric links
 - Coherent memory across the node
 - 100 GB/s injection bandwidth
 - Near-node NVM storage



Comparison of Titan, Summit, and Frontier Systems

System Specs	Titan	Summit	Frontier
Peak	27 PF	200 PF	~1.5 EF
# cabinets	200	256	> 100
Node	1 AMD Opteron CPU 1 NVIDIA Kepler GPU	2 IBM POWER9™ CPUs 6 NVIDIA Volta GPUs	1 AMD EPYC CPU 4 AMD Radeon Instinct GPUs
On-node interconnect	PCI Gen2 No coherence across the node	NVIDIA NVLINK Coherent memory across the node	AMD Infinity Fabric Coherent memory across the node
System Interconnect	Cray Gemini network 6.4 GB/s	Mellanox Dual-port EDR IB network 25 GB/s	Cray four-port Slingshot network 100 GB/s
Topology	3D Torus	Non-blocking Fat Tree	Dragonfly
Storage	32 PB, 1 TB/s, Lustre Filesystem	250 PB, 2.5 TB/s, IBM Spectrum Scale™ with GPFS™	2-4x performance and capacity of Summit's I/O subsystem.
On-node NVM	No	Yes	Yes
Power	9 MV	13 MV	29 MV

Complex architectures yield...



Complex Programming Models

During this Sixth Wave transition, **Complexity** is our major challenge!

Design

How do we design future systems so that they are better than current systems on important applications?

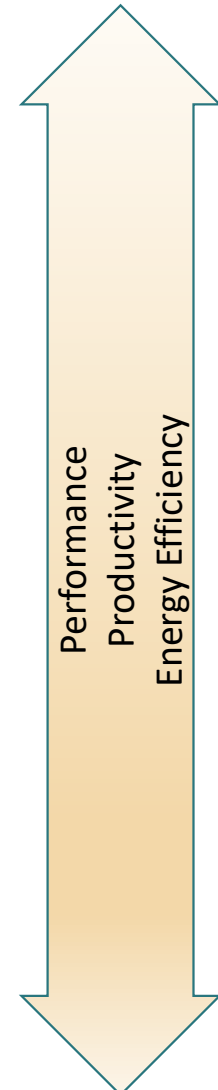
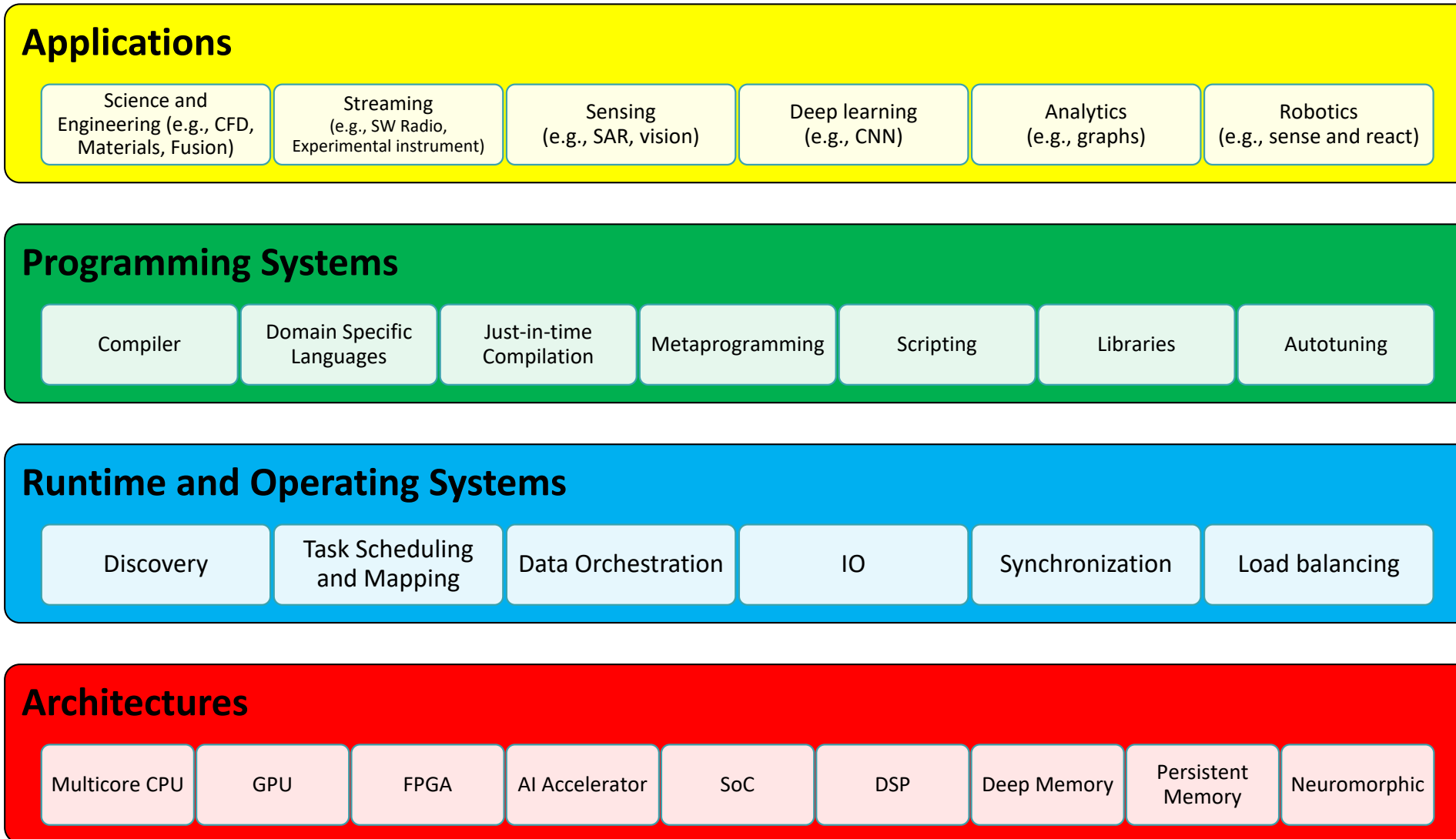
- Simulation and modeling are more difficult
- Entirely possible that the new system will be slower than the old system!
- Expect 'disaster' procurements

Programmability

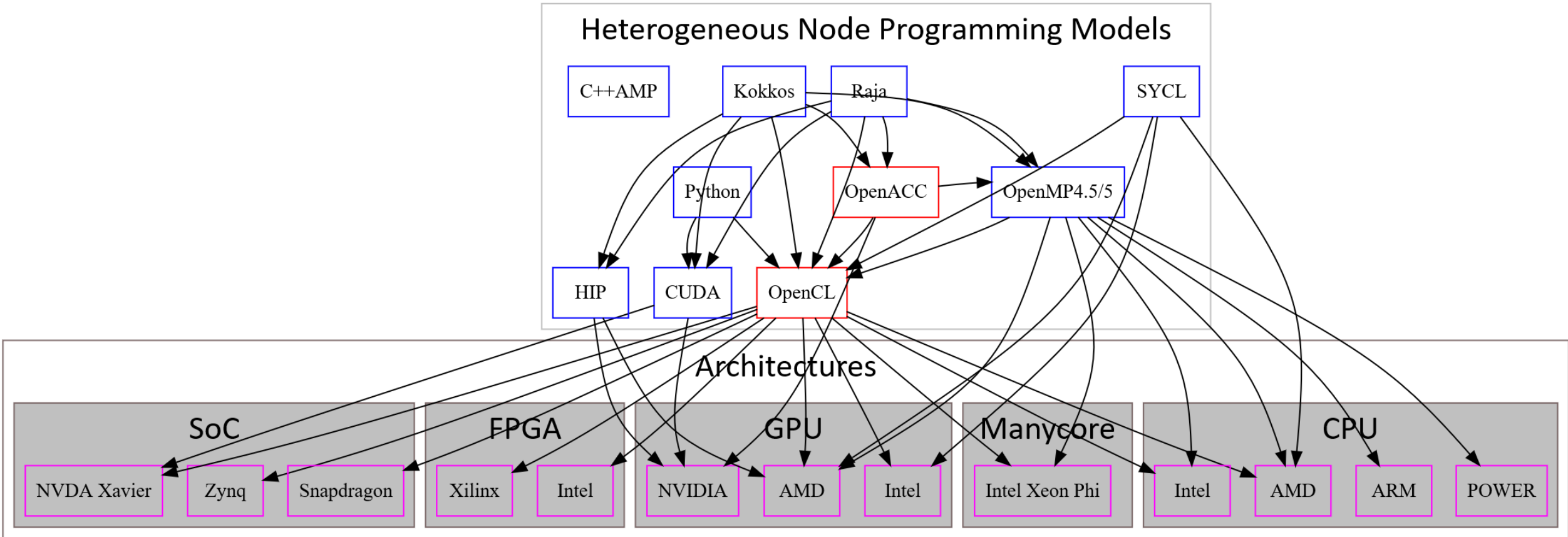
How do we design applications with some level of performance portability?

- Software lasts much longer than transient hardware platforms
- Proper abstractions for flexibility and efficiency
- Adapt or die

The FTG Vision | Programming Systems

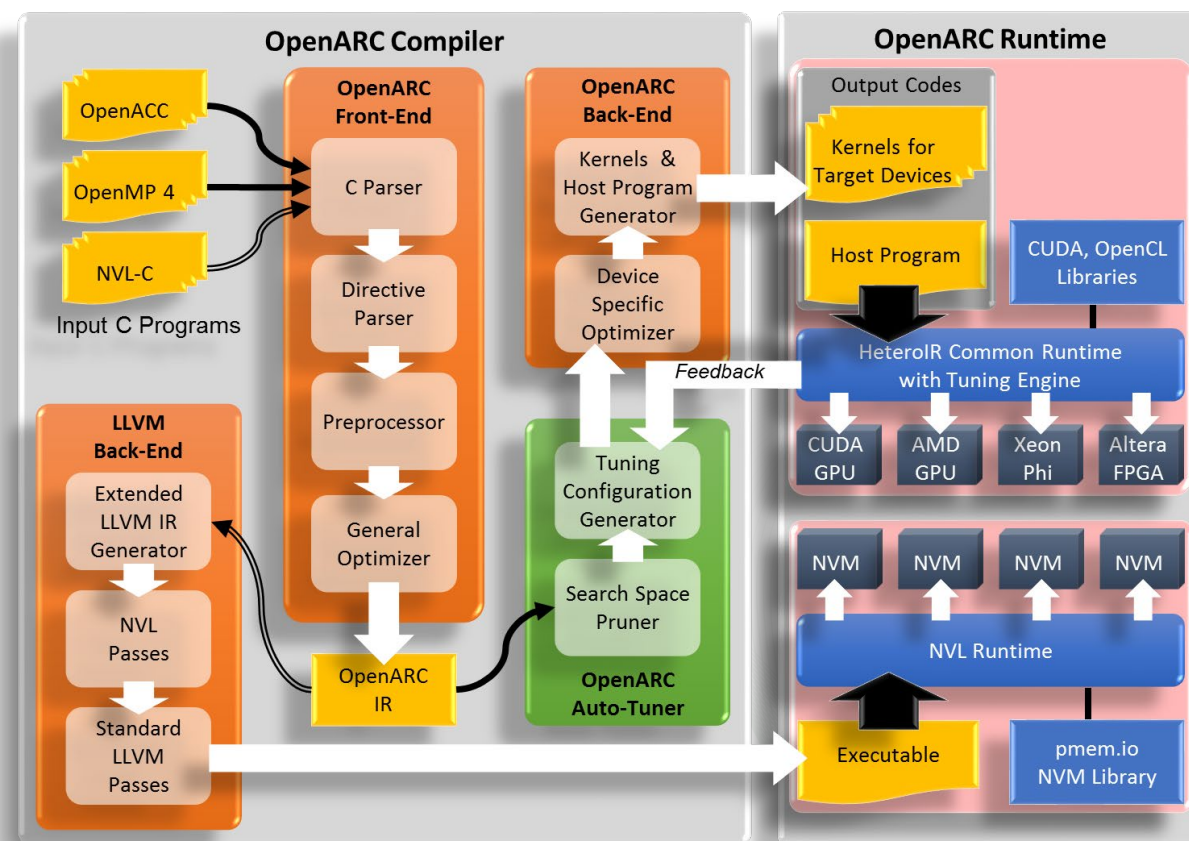


What more to say ?!?!? 😊



Directive-based Strategy with OpenARC: Open Accelerator Research Compiler

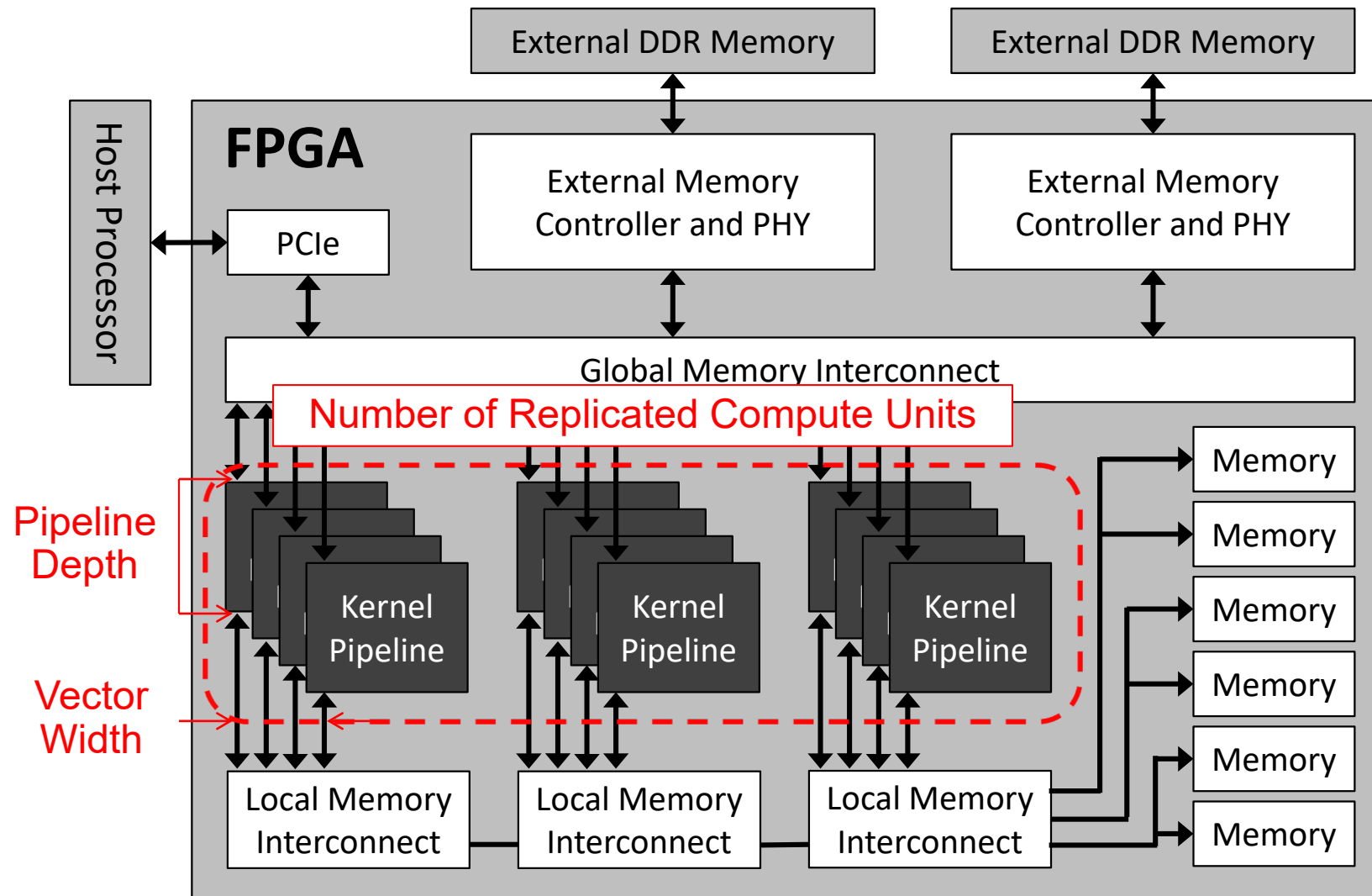
- Open-Sourced, High-Level Intermediate Representation (HIR)-Based, Extensible Compiler Framework.
 - Perform source-to-source translation from OpenACC C to target accelerator models.
 - Support full features of OpenACC V1.0 (+ array reductions and function calls)
 - Support both CUDA and OpenCL as target accelerator models
 - Provide common runtime APIs for various back-ends
 - Can be used as a research framework for various study on directive-based accelerator computing.
 - Built on top of Cetus compiler framework, equipped with various advanced analysis/transformation passes and built-in tuning tools.
 - OpenARC's IR provides an AST-like syntactic view of the source program, easy to understand, access, and transform the input program.



FPGAs | Approach

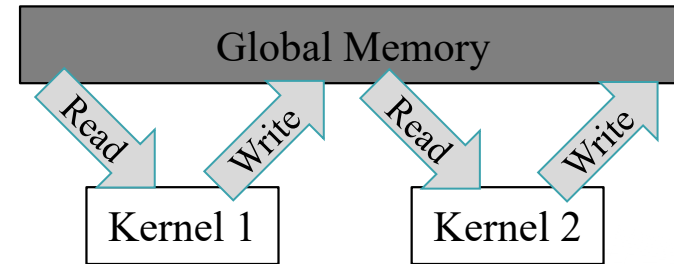
- Design and implement an OpenACC-to-FPGA translation framework, which is the first work to use a standard and portable directive-based, high-level programming system for FPGAs.
- Propose FPGA-specific optimizations and novel pragma extensions to improve performance.
- Evaluate the functional and performance portability of the framework across diverse architectures (Altera FPGA, NVIDIA GPU, AMD GPU, and Intel Xeon Phi).

FPGA OpenCL Architecture

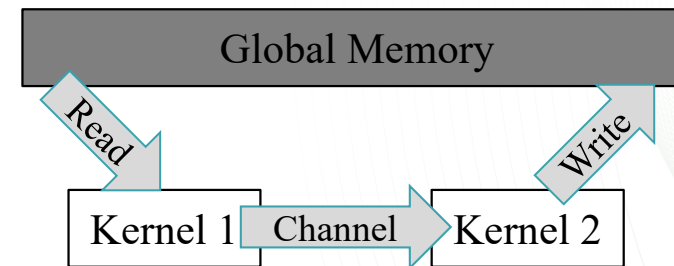


Kernel-Pipelining Transformation Optimization

- Kernel execution model in OpenACC
 - Device kernels can communicate with each other only through the device global memory.
 - Synchronizations between kernels are at the granularity of a kernel execution.
- Altera OpenCL channels
 - Allows passing data between kernels and synchronizing kernels with high efficiency and low latency



Kernel communications through global memory in OpenACC



Kernel communications with Altera channels

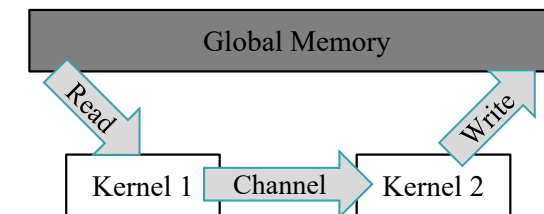
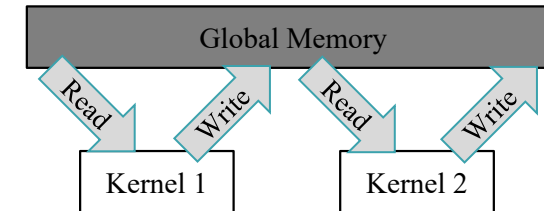
Kernel-Pipelining Transformation Optimization (2)

(a) Input OpenACC code

```
#pragma acc data copyin (a) create (b) copyout (c)
{
  #pragma acc kernels loop gang worker present (a, b)
  for(i=0; i<N; i++) { b[i] = a[i]*a[i]; }
  #pragma acc kernels loop gang worker present (b, c)
  for(i=0; i<N; i++) { c[i] = b[i]; }
}
```

(b) Altera OpenCL code with channels

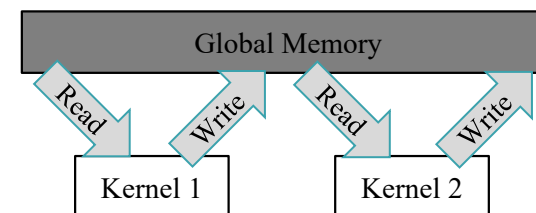
```
channel float pipe_b;
__kernel void kernel1(__global float* a) {
  int i = get_global_id(0);
  write_channel_altera(pipe_b, a[i]*a[i]);
}
__kernel void kernel2(__global float* c) {
  int i = get_global_id(0);
  c[i] = read_channel_altera(pipe_b);
}
```



Kernel-Pipelining Transformation Optimization (3)

(a) Input OpenACC code

```
#pragma acc data copyin (a) create (b) copyout (c)
{
  #pragma acc kernels loop gang worker present (a, b)
  for(i=0; i<N; i++) { b[i] = a[i]*a[i]; }
  #pragma acc kernels loop gang worker present (b, c)
  for(i=0; i<N; i++) {c[i] = b[i]; }
}
```



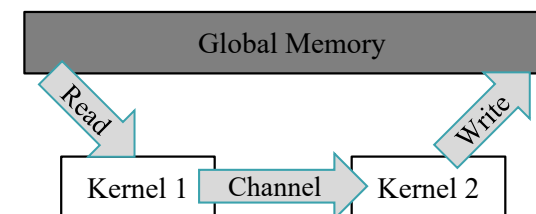
Kernel-pipelining transformation

Valid under specific conditions



(c) Modified OpenACC code for kernel-pipelining

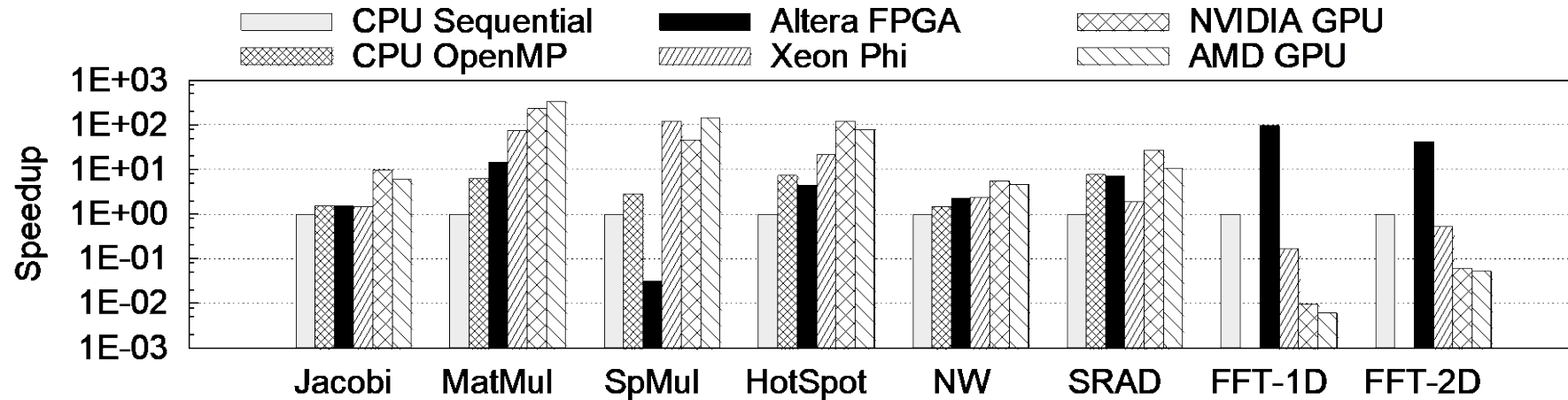
```
#pragma acc data copyin (a) pipe (b) copyout (c)
{
  #pragma acc kernels loop gang worker pipeout (b) present (a)
  For(i=0; i<N; i++) { b[i] = a[i]*a[i]; }
  #pragma acc kernels loop gang worker pipein (b) present (c)
  For(i=0; i<N; i++) {c[i] = b[i];}
}
```



FPGA-specific Optimizations

- Single work-item
- Collapse
- Reduction
- Sliding window
- (Branch-variant code motion)
- (Custom unrolling)

Overall Performance of OpenARC FPGA Evaluation



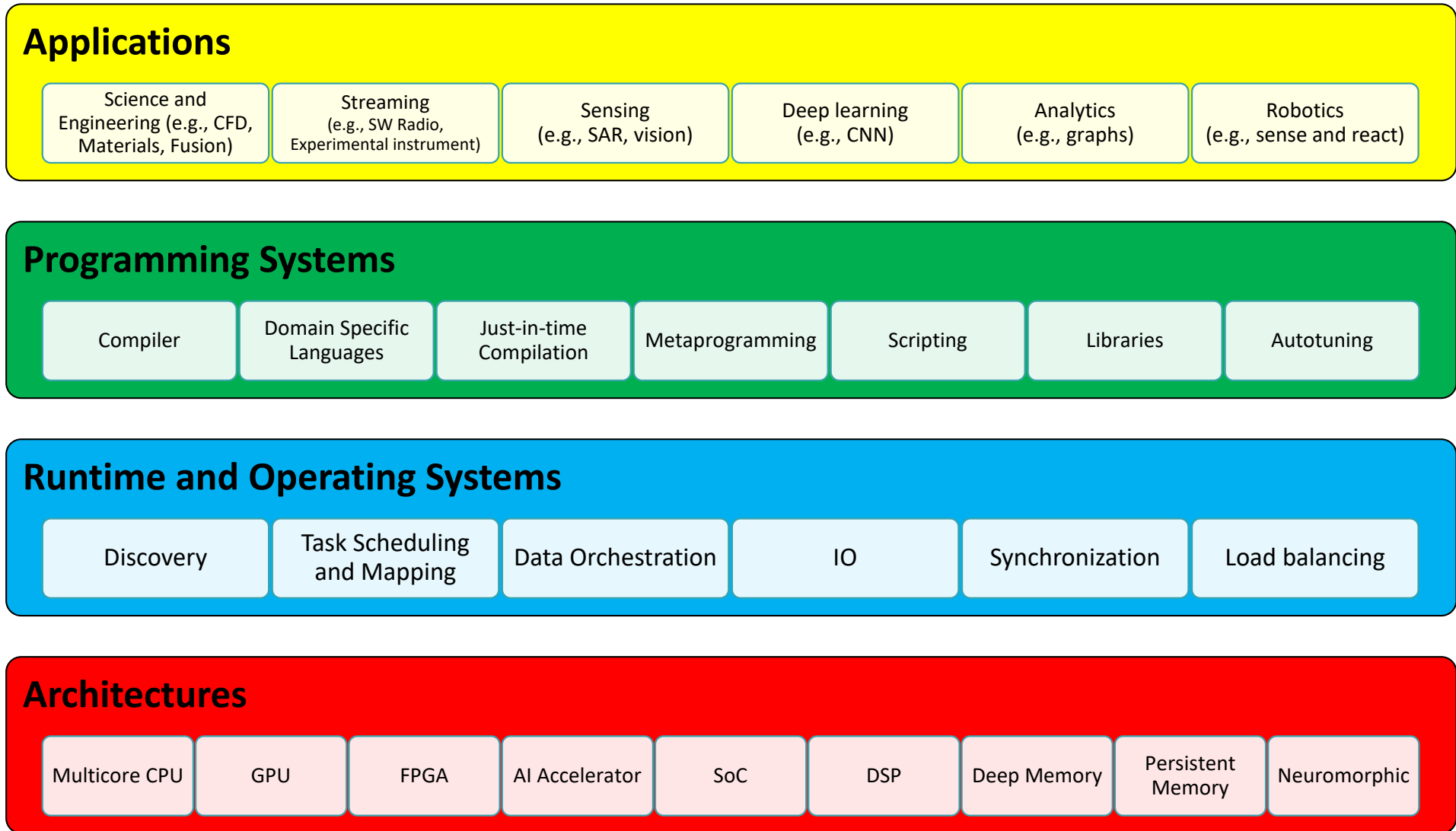
FPGAs prefer applications with deep execution pipelines (e.g., FFT-1D and FFT-2D), performing much higher than other accelerators.

For traditional HPC applications with abundant parallel floating-point operations, it seems to be difficult for FPGAs to beat the performance of other accelerators, even though FPGAs can be much more power-efficient.

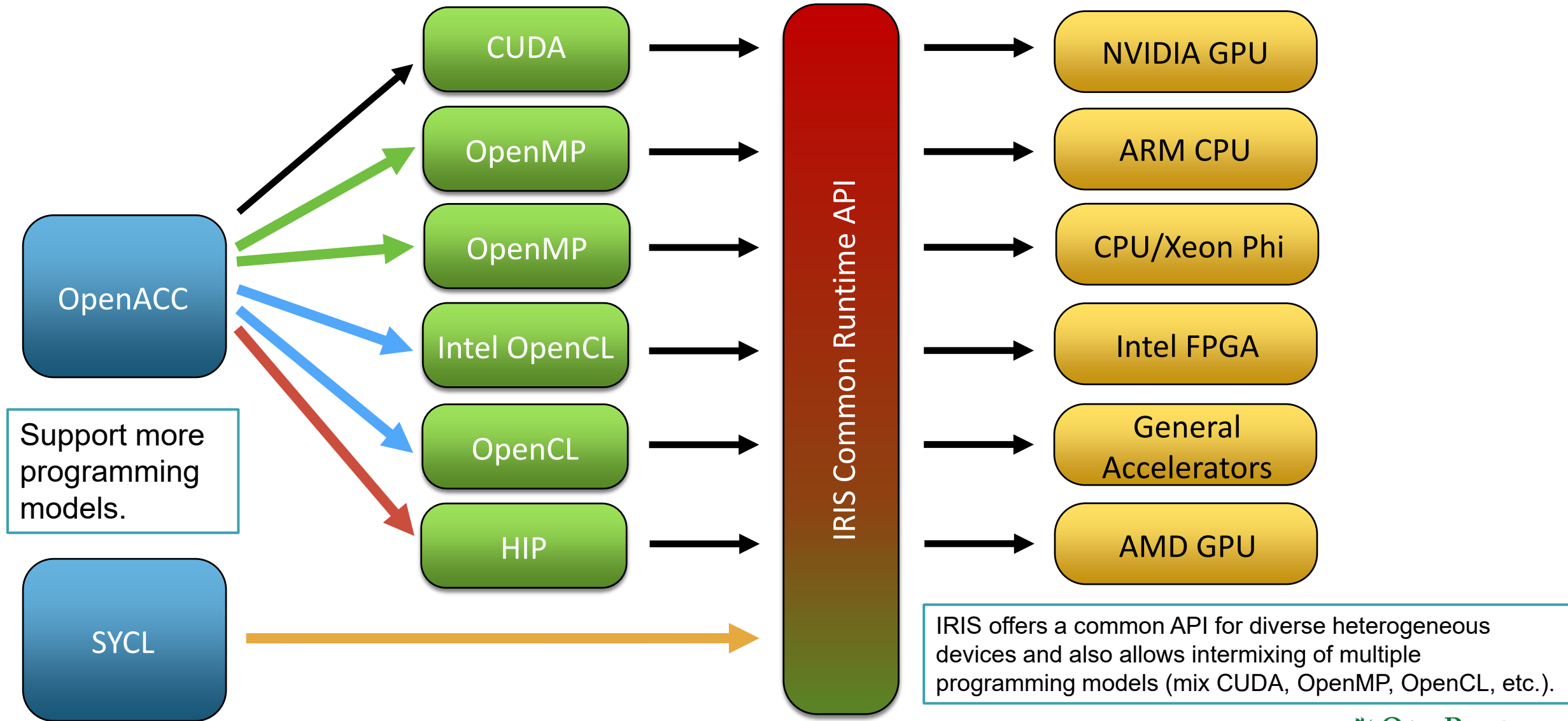
- Tested FPGA does not contain dedicated, embedded floating-point cores, while others have fully-optimized floating-point computation units.

Current and upcoming high-end FPGAs are equipped with hardened floating-point operators, whose performance will be comparable to other accelerators, while remaining power-efficient.

The FTG Vision | Runtime and Operating Systems

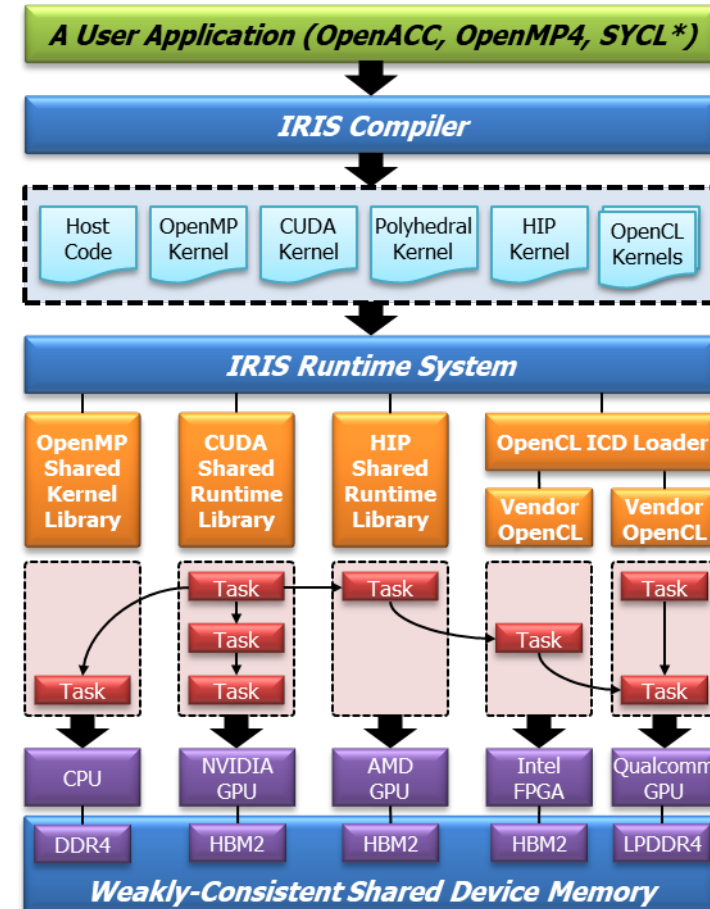


IRIS: Mapping Strategy for Heterogeneous Architectures *and Native Programming Models*



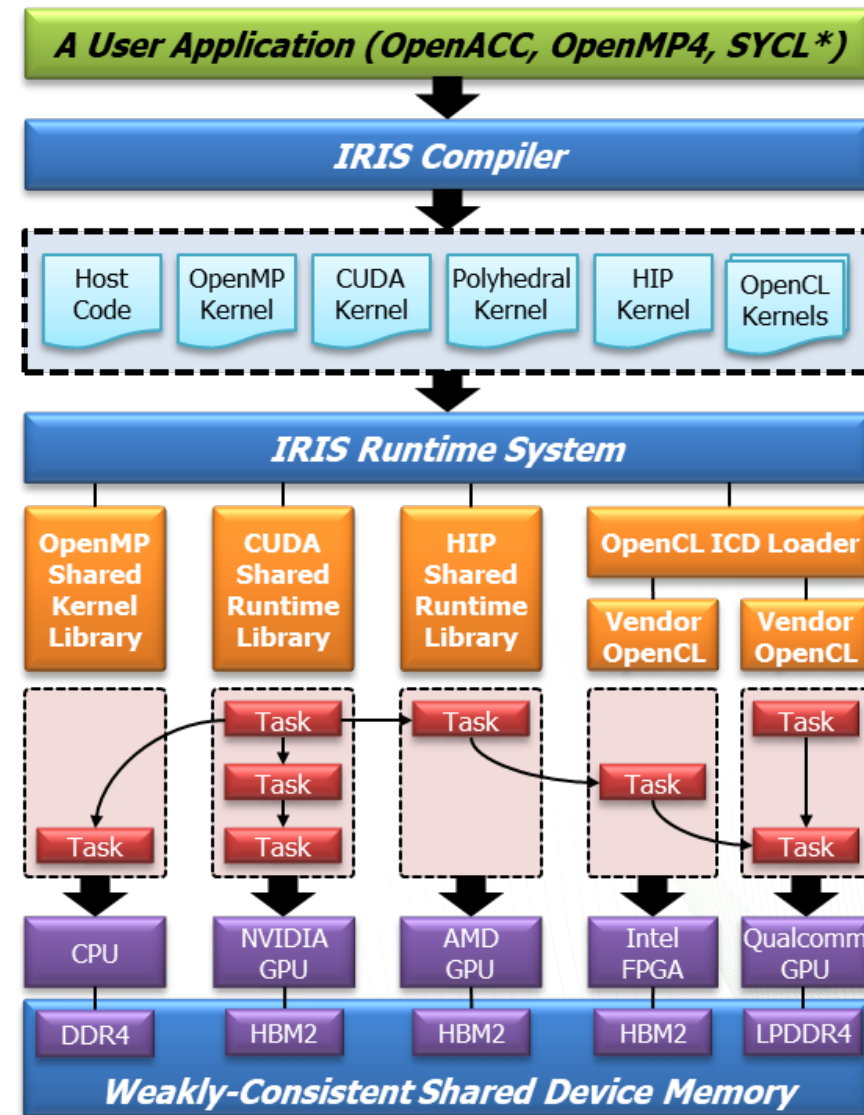
IRIS: An Intelligent Runtime System for Extremely Heterogeneous Architectures

- Provide programmers a unified programming environment to write portable code across heterogeneous architectures (and preferred programming systems)
- Orchestrate diverse programming systems (OpenCL, CUDA, HIP, OpenMP for CPU) in a single application
 - OpenCL
 - NVIDIA GPU, AMD GPU, ARM GPU, Qualcomm GPU, Intel CPU, Intel Xeon Phi, Intel FPGA, Xilinx FPGA
 - CUDA
 - NVIDIA GPU
 - HIP
 - AMD GPU
 - OpenMP for CPU
 - Intel CPU, AMD CPU, PowerPC CPU, ARM CPU, Qualcomm CPU



The IRIS Architecture

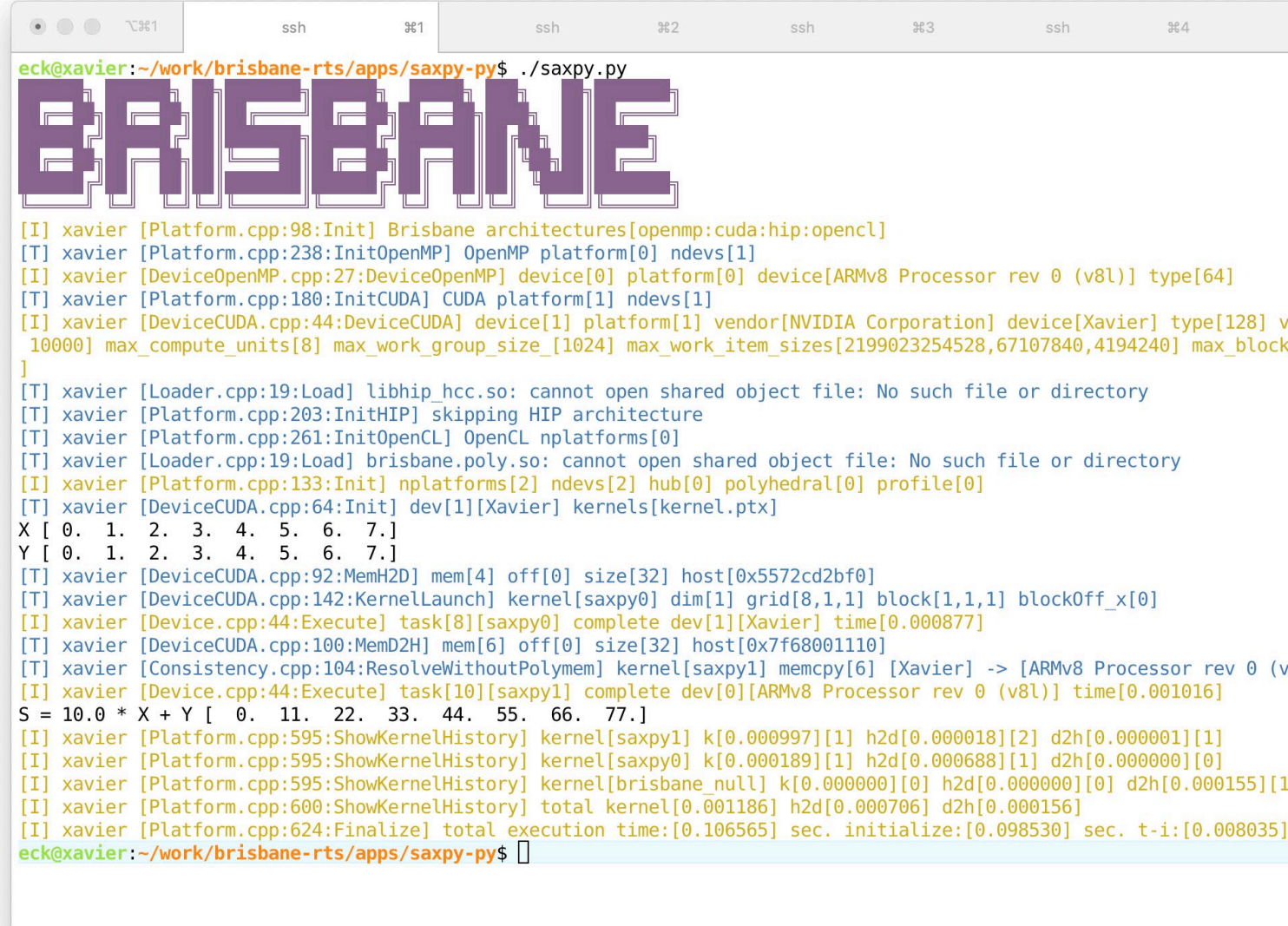
- Platform Model
 - A single-node system equipped with host CPUs and multiple compute devices (GPUs, FPGAs, Xeon Phis, and multicore CPUs)
- Memory Model
 - Host memory + shared device memory
 - All compute devices share the device memory
- Execution Model
 - DAG-style task parallel execution across all available compute devices
- Programming Model
 - High-level OpenACC, OpenMP4, SYCL* (* planned)
 - Low-level C/Fortran/Python IRIS host-side runtime API + OpenCL/CUDA/HIP/OpenMP kernels (w/o compiler support)



Supported Architectures and Programming Systems by IRIS

ExCL* Systems	Oswald	Summit-node	Radeon	Xavier	Snapdragon
CPU	Intel Xeon	IBM Power9	Intel Xeon	ARMv8	Qualcomm Kryo
Programming Systems	<ul style="list-style-type: none"> Intel OpenMP Intel OpenCL 	<ul style="list-style-type: none"> IBM XL OpenMP 	<ul style="list-style-type: none"> Intel OpenMP Intel OpenCL 	<ul style="list-style-type: none"> GNU GOMP 	<ul style="list-style-type: none"> Android NDK OpenMP
GPU	NVIDIA P100	NVIDIA V100	AMD Radeon VII	NVIDIA Volta	Qualcomm Adreno 640
Programming Systems	<ul style="list-style-type: none"> NVIDIA CUDA NVIDIA OpenCL 	<ul style="list-style-type: none"> NVIDIA CUDA 	<ul style="list-style-type: none"> AMD HIP AMD OpenCL 	<ul style="list-style-type: none"> NVIDIA CUDA 	<ul style="list-style-type: none"> Qualcomm OpenCL
FPGA	Intel/Altera Stratix 10				
Programming Systems	<ul style="list-style-type: none"> Intel OpenCL 				

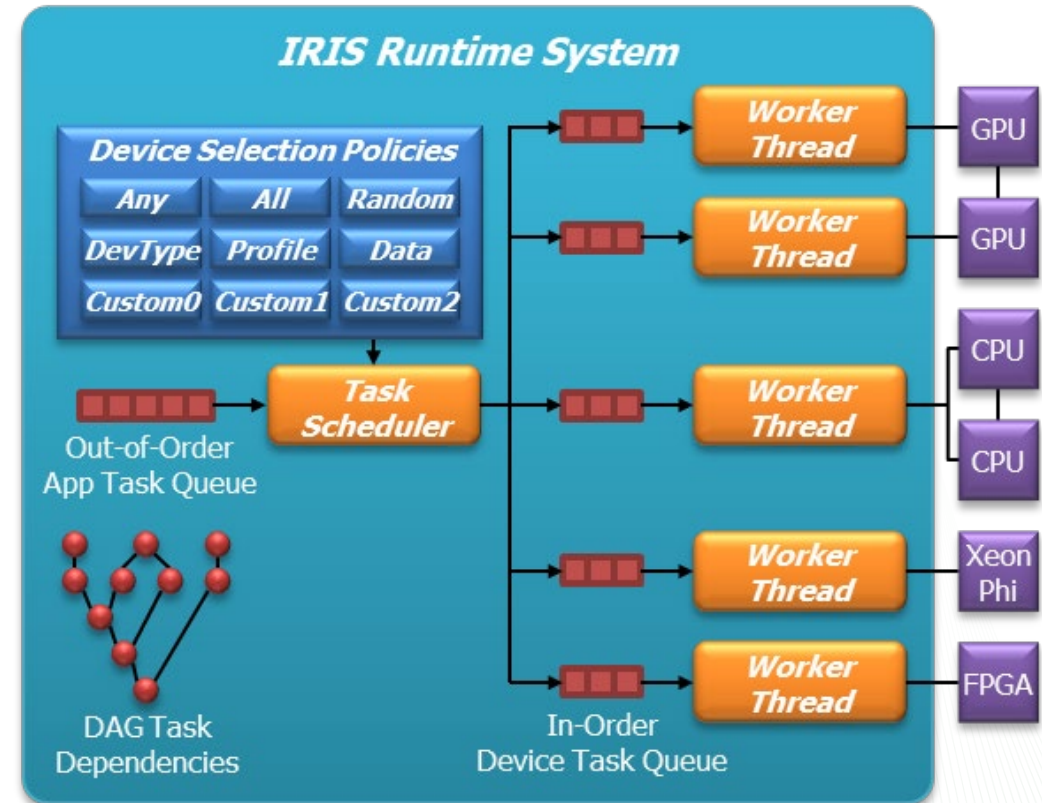
IRIS Booting on Various Platforms



```
[I] xavier [Platform.cpp:98:Init] Brisbane architectures[openmp:cuda:hip:opencl]
[T] xavier [Platform.cpp:238:InitOpenMP] OpenMP platform[0] ndevs[1]
[I] xavier [DeviceOpenMP.cpp:27:DeviceOpenMP] device[0] platform[0] device[ARMv8 Processor rev 0 (v8l)] type[64]
[T] xavier [Platform.cpp:180:InitCUDA] CUDA platform[1] ndevs[1]
[I] xavier [DeviceCUDA.cpp:44:DeviceCUDA] device[1] platform[1] vendor[NVIDIA Corporation] device[Xavier] type[128] vendor[10000] max_compute_units[8] max_work_group_size[1024] max_work_item_sizes[2199023254528,67107840,4194240] max_block_size[1024]
[T] xavier [Loader.cpp:19:Load] libhip_hcc.so: cannot open shared object file: No such file or directory
[T] xavier [Platform.cpp:203:InitHIP] skipping HIP architecture
[T] xavier [Platform.cpp:261:InitOpenCL] OpenCL nplatforms[0]
[T] xavier [Loader.cpp:19:Load] brisbane.poly.so: cannot open shared object file: No such file or directory
[I] xavier [Platform.cpp:133:Init] nplatforms[2] ndevs[2] hub[0] polyhedral[0] profile[0]
[T] xavier [DeviceCUDA.cpp:64:Init] dev[1][Xavier] kernels[kernel.ptx]
X [ 0. 1. 2. 3. 4. 5. 6. 7.]
Y [ 0. 1. 2. 3. 4. 5. 6. 7.]
[T] xavier [DeviceCUDA.cpp:92:MemH2D] mem[4] off[0] size[32] host[0x5572cd2bf0]
[T] xavier [DeviceCUDA.cpp:142:KernelLaunch] kernel[saxpy0] dim[1] grid[8,1,1] block[1,1,1] blockOff_x[0]
[I] xavier [Device.cpp:44:Execute] task[8][saxpy0] complete dev[1][Xavier] time[0.000877]
[T] xavier [DeviceCUDA.cpp:100:MemD2H] mem[6] off[0] size[32] host[0x7f68001110]
[T] xavier [Consistency.cpp:104:ResolveWithoutPolymem] kernel[saxpy1] memcpy[6] [Xavier] -> [ARMv8 Processor rev 0 (v8l)]
[I] xavier [Device.cpp:44:Execute] task[10][saxpy1] complete dev[0][ARMv8 Processor rev 0 (v8l)] time[0.001016]
S = 10.0 * X + Y [ 0. 11. 22. 33. 44. 55. 66. 77.]
[I] xavier [Platform.cpp:595:ShowKernelHistory] kernel[saxpy1] k[0.000997][1] h2d[0.000018][2] d2h[0.000000][1]
[I] xavier [Platform.cpp:595:ShowKernelHistory] kernel[saxpy0] k[0.000189][1] h2d[0.000688][1] d2h[0.000000][0]
[I] xavier [Platform.cpp:595:ShowKernelHistory] kernel[brisbane_null] k[0.000000][0] h2d[0.000000][0] d2h[0.000155][1]
[I] xavier [Platform.cpp:600:ShowKernelHistory] total kernel[0.001186] h2d[0.000706] d2h[0.000156]
[I] xavier [Platform.cpp:624:Finalize] total execution time:[0.106565] sec. initialize:[0.098530] sec. t-i:[0.008035]
eck@xavier:~/work/brisbane-rts/apps/saxpy-py$
```

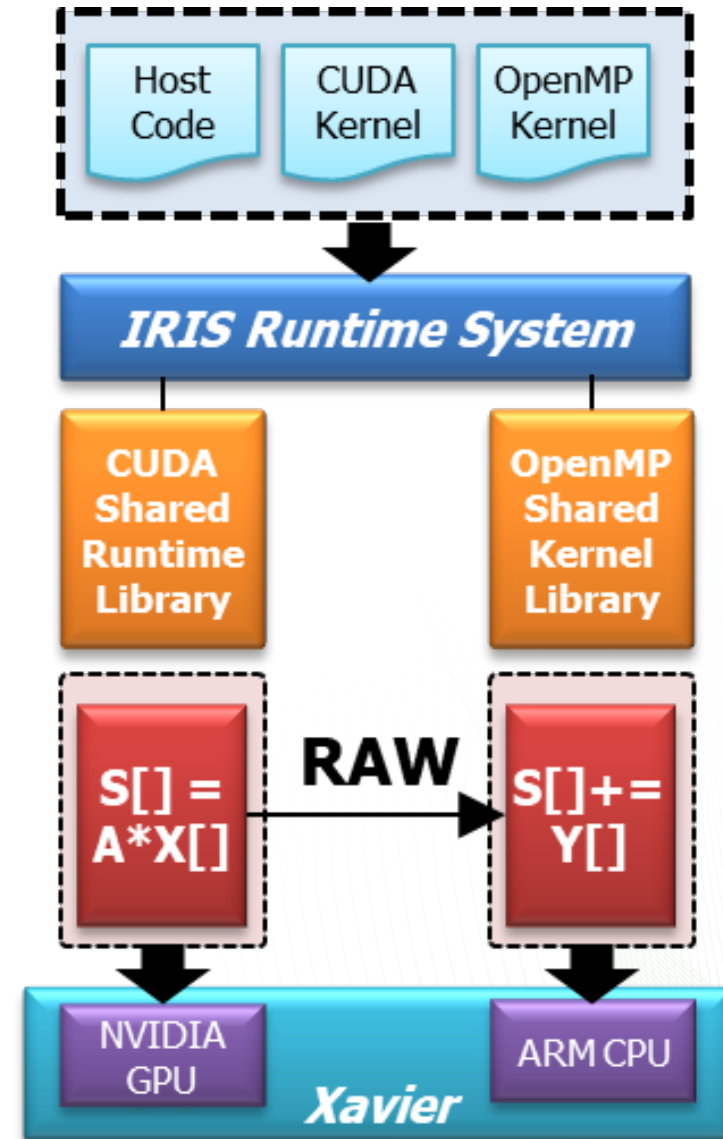
Task Scheduling in IRIS

- A task
 - A scheduling unit
 - Contains multiple in-order commands
 - Kernel launch command
 - Memory copy command (device-to-host, host-to-device)
 - May have DAG-style dependencies with other tasks
 - Enqueued to the application task queue with a device selection policy
 - Available device selection policies
 - Specific Device (compute device #)
 - Device Type (CPU, GPU, FPGA, XeonPhi)
 - Profile-based
 - Locality-aware
 - Ontology-base
 - Performance models (Aspen)
 - Any, All, Random, 3rd-party users' custom policies
- The task scheduler dispatches the tasks in the application task queue to available compute devices
 - Select the optimal target compute device according to task's device selection policy



SAXPY Example on Xavier

- Computation
 - $S[] = A * X[] + Y[]$
- Two tasks
 - $S[] = A * X[]$ on NVIDIA GPU (CUDA)
 - $S[] += Y[]$ on ARM CPU (OpenMP)
 - $S[]$ is shared between two tasks
 - Read-after-write (RAW), true dependency
- Low-level Python IRIS host code + CUDA/OpenMP kernels
 - saxpy.py
 - kernel.cu
 - kernel.openmp.h



SAXPY: Python host code & CUDA kernel code

saxpy.py (1/2)

```
#!/usr/bin/env python

import iris
import numpy as np
import sys

iris.init()

SIZE = 1024
A = 10.0

x = np.arange(SIZE, dtype=np.float32)
y = np.arange(SIZE, dtype=np.float32)
s = np.arange(SIZE, dtype=np.float32)

print 'X', x
print 'Y', y

mem_x = iris.mem(x.nbytes)
mem_y = iris.mem(y.nbytes)
mem_s = iris.mem(s.nbytes)
```

saxpy.py (2/2)

```
kernel0 = iris.kernel("saxpy0")
kernel0.setmem(0, mem_s, iris.iris_w)
kernel0.setint(1, A)
kernel0.setmem(2, mem_x, iris.iris_r)

off = [ 0 ]
ndr = [ SIZE ]

task0 = iris.task()
task0.h2d_full(mem_x, x)
task0.kernel(kernel0, 1, off, ndr)
task0.submit(iris.iris_gpu)

kernel1 = iris.kernel("saxpy1")
kernel1.setmem(0, mem_s, iris.iris_rw)
kernel1.setmem(1, mem_y, iris.iris_r)

task1 = iris.task()
task1.h2d_full(mem_y, y)
task1.kernel(kernel1, 1, off, ndr)
task1.d2h_full(mem_s, s)
task1.submit(iris.iris_cpu)

print 'S =', A, '* X + Y', s

iris.finalize()
```

kernel.cu (CUDA)

```
extern "C" __global__ void saxpy0(float* S, float
A, float* X) {
    int id = blockIdx.x * blockDim.x + threadIdx.x;
    S[id] = A * X[id];
}

extern "C" __global__ void saxpy1(float* S,
float* Y) {
    int id = blockIdx.x * blockDim.x + threadIdx.x;
    S[id] += Y[id];
}
```

SAXPY: Python host code & OpenMP kernel code

saxpy.py (1/2)

```
#!/usr/bin/env python

import iris
import numpy as np
import sys

iris.init()

SIZE = 1024
A = 10.0

x = np.arange(SIZE, dtype=np.float32)
y = np.arange(SIZE, dtype=np.float32)
s = np.arange(SIZE, dtype=np.float32)

print 'X', x
print 'Y', y

mem_x = iris.mem(x.nbytes)
mem_y = iris.mem(y.nbytes)
mem_s = iris.mem(s.nbytes)
```

saxpy.py (2/2)

```
kernel0 = iris.kernel("saxpy0")
kernel0.setmem(0, mem_s, iris.iris_w)
kernel0.setint(1, A)
kernel0.setmem(2, mem_x, iris.iris_r)

off = [ 0 ]
ndr = [ SIZE ]

task0 = iris.task()
task0.h2d_full(mem_x, x)
task0.kernel(kernel0, 1, off, ndr)
task0.submit(iris.iris_gpu)

kernel1 = iris.kernel("saxpy1")
kernel1.setmem(0, mem_s, iris.iris_rw)
kernel1.setmem(1, mem_y, iris.iris_r)

task1 = iris.task()
task1.h2d_full(mem_y, y)
task1.kernel(kernel1, 1, off, ndr)
task1.d2h_full(mem_s, s)
task1.submit(iris.iris_cpu)

print 'S =', A, '* X + Y', s

iris.finalize()
```

kernel.openmp.h (OpenMP)

```
#include <iris/iris_openmp.h>

static void saxpy0(float* S, float A, float* X,
IRIS_OPENMP_KERNEL_ARGS) {
    int id;
    #pragma omp parallel for shared(S, A, X)
    private(id)
        IRIS_OPENMP_KERNEL_BEGIN
        S[id] = A * X[id];
        IRIS_OPENMP_KERNEL_END
    }

static void saxpy1(float* S, float* Y,
IRIS_OPENMP_KERNEL_ARGS) {
    int id;
    #pragma omp parallel for shared(S, Y) private(id)
        IRIS_OPENMP_KERNEL_BEGIN
        S[id] += Y[id];
        IRIS_OPENMP_KERNEL_END
    }
```

Memory Consistency Management

saxpy.py (1/2)

```
#!/usr/bin/env python

import iris
import numpy as np
import sys

iris.init()

SIZE = 1024
A = 10.0

x = np.arange(SIZE, dtype=np.float32)
y = np.arange(SIZE, dtype=np.float32)
s = np.arange(SIZE, dtype=np.float32)

print 'X', x
print 'Y', y

mem_x = iris.mem(x.nbytes)
mem_y = iris.mem(y.nbytes)
mem_s = iris.mem(s.nbytes)
```

saxpy.py (2/2)

```
kernel0 = iris.kernel("saxpy0")
kernel0.setmem(0, mem_s, iris.iris_w)
kernel0.setint(1, A)
kernel0.setmem(2, mem_x, iris.iris_r)

off = [ 0 ]
ndr = [ SIZE ]

task0 = iris.task()
task0.h2d_full(mem_x, x)
task0.kernel(kernel0, 1, off, ndr)
task0.submit(iris.iris_gpu)

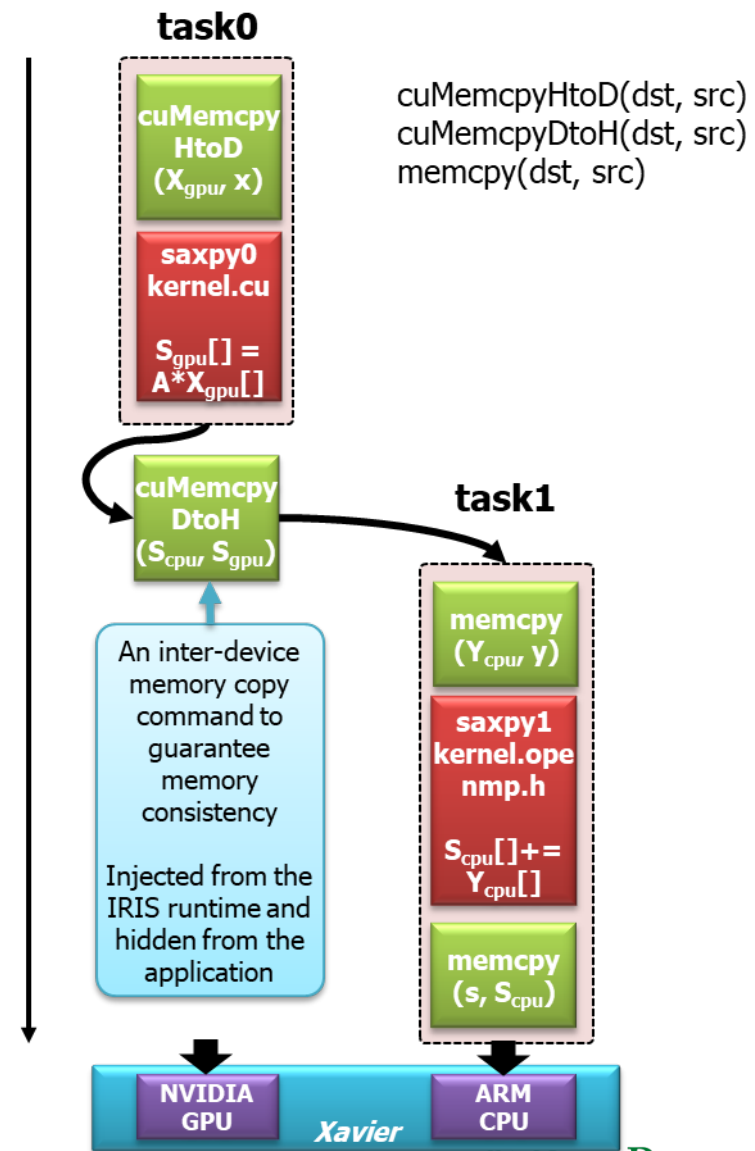
kernel1 = iris.kernel("saxpy1")
kernel1.setmem(0, mem_s, iris.iris_rw)
kernel1.setmem(1, mem_y, iris.iris_r)

task1 = iris.task()
task1.h2d_full(mem_y, y)
task1.kernel(kernel1, 1, off, ndr)
task1.d2h_full(mem_s, s)
task1.submit(iris.iris_cpu)

print 'S =', A, '* X + Y', s

iris.finalize()
```

mem_s is shared between GPU and CPU



Locality-aware Device Selection Policy

saxpy.py (1/2)

```
#!/usr/bin/env python

import iris
import numpy as np
import sys

iris.init()

SIZE = 1024
A = 10.0

x = np.arange(SIZE, dtype=np.float32)
y = np.arange(SIZE, dtype=np.float32)
s = np.arange(SIZE, dtype=np.float32)

print 'X', x
print 'Y', y

mem_x = iris.mem(x.nbytes)
mem_y = iris.mem(y.nbytes)
mem_s = iris.mem(s.nbytes)
```

saxpy.py (2/2)

```
kernel0 = iris.kernel("saxpy0")
kernel0.setmem(0, mem_s, iris.iris_w)
kernel0.setint(1, A)
kernel0.setmem(2, mem_x, iris.iris_r)

off = [ 0 ]
ndr = [ SIZE ]

task0 = iris.task()
task0.h2d_full(mem_x, x)
task0.kernel(kernel0, 1, off, ndr)
task0.submit(iris.iris_gpu)

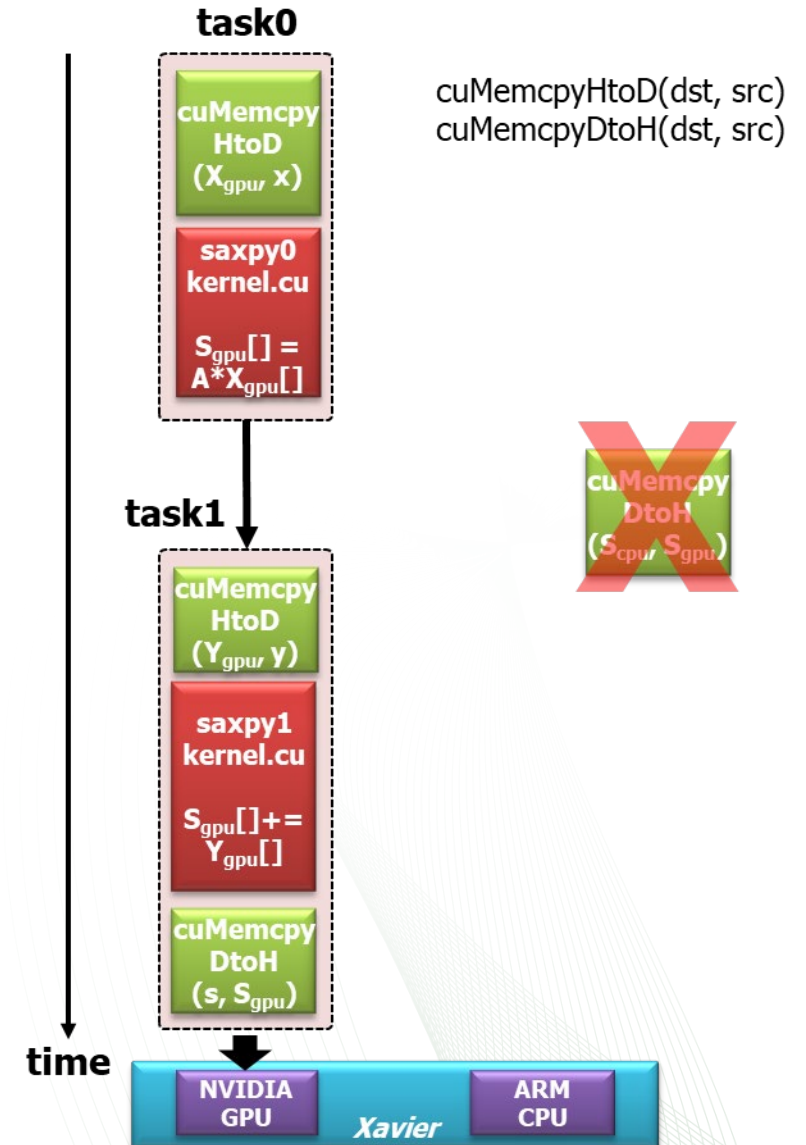
kernel1 = iris.kernel("saxpy1")
kernel1.setmem(0, mem_s, iris.iris_rw)
kernel1.setmem(1, mem_y, iris.iris_r)

task1 = iris.task()
task1.h2d_full(mem_y, y)
task1.kernel(kernel1, 1, off, ndr)
task1.d2h_full(mem_s, s)
task1.submit(iris.iris_data)

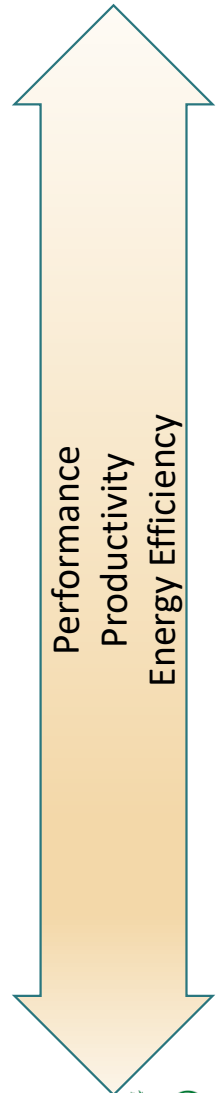
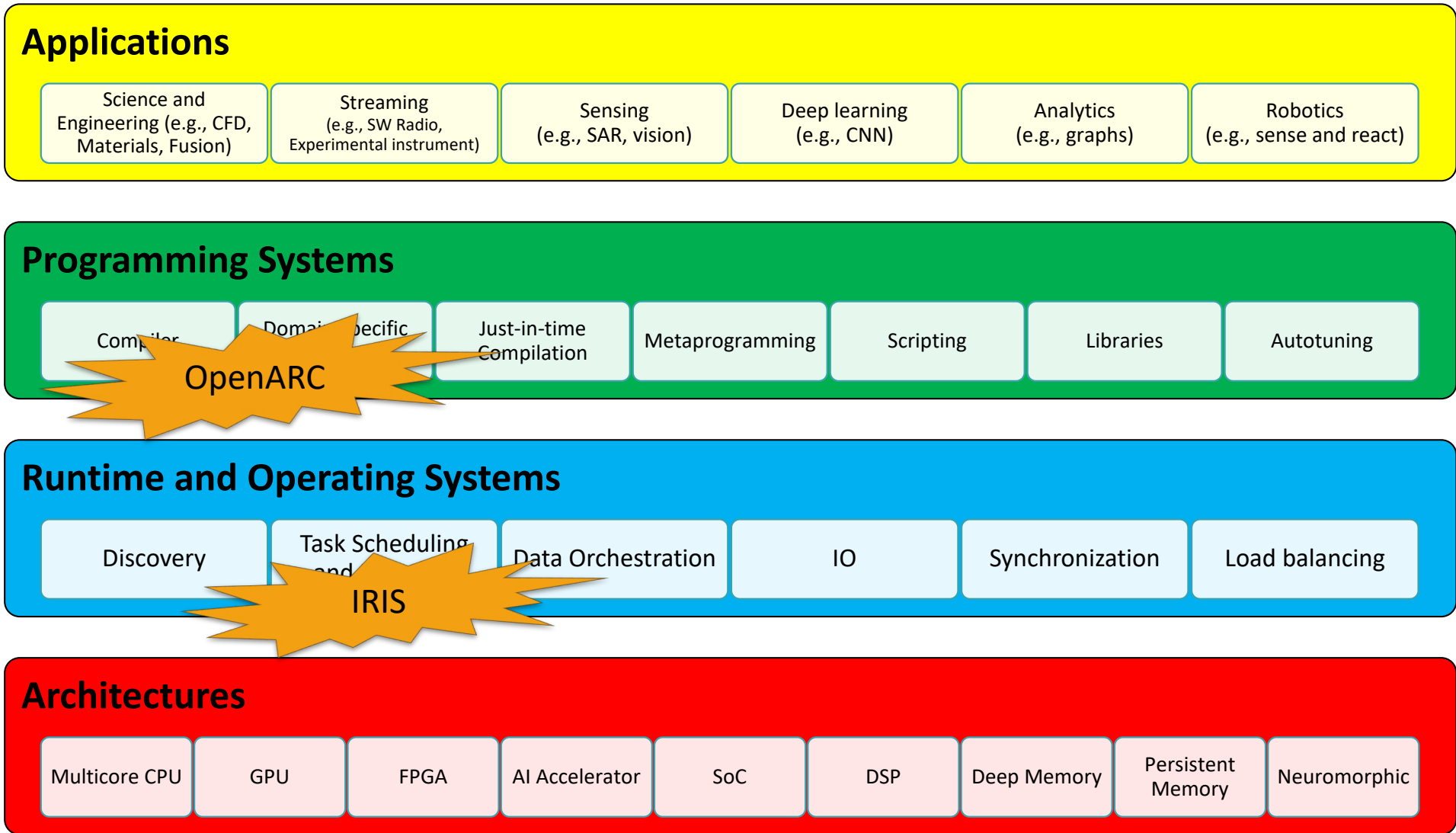
print 'S =', A, '* X + Y', s

iris.finalize()
```

iris_data selects the device that requires minimum data transfer to execute the task



The FTG Vision



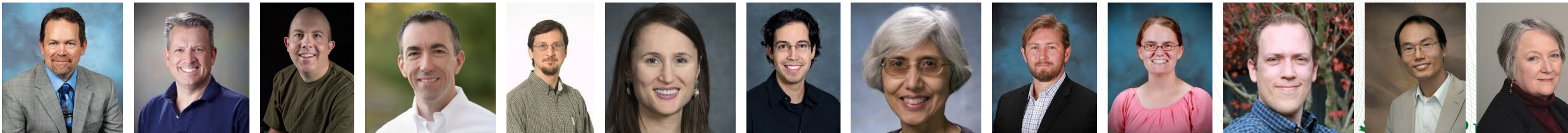
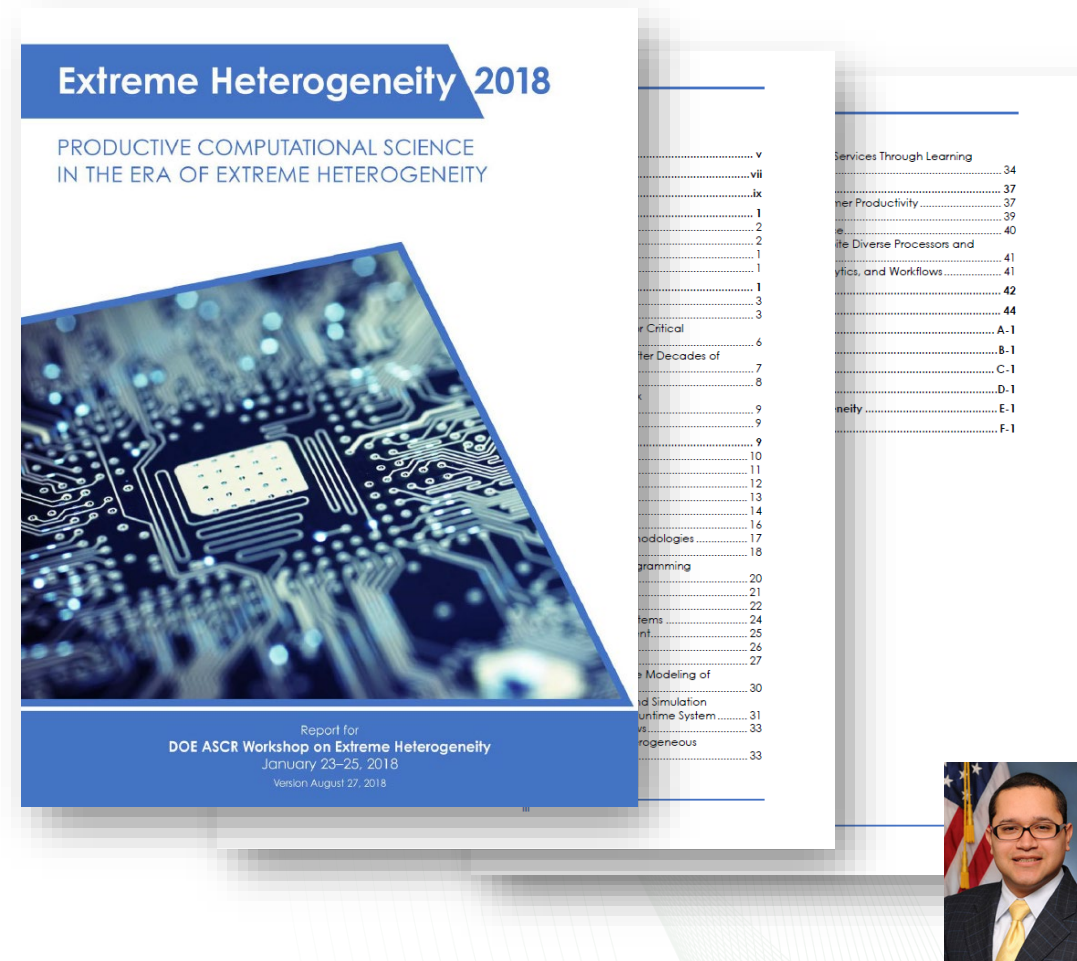
Recap

- Motivation: Recent trends in computing paint an ambiguous future
 - Multiple architectural dimensions are being (dramatically) redesigned: Processors, node design, memory systems, I/O
 - Complexity is our main challenge
- Applications and software systems across many areas are all reaching a state of crisis
 - Need a focus on performance portability
- ORNL FTG investigating design and programming challenges for these trends
 - Performance modeling and ontologies
 - Performance portable compilation to many different heterogeneous architectures/SoCs
 - Intelligent scheduling system to automate discovery, device selection, and data movement
 - Targeting wide variety of existing and future architectures (DSSoC and others)

- Visit us
 - We host interns and other visitors year round
 - Faculty, grad, undergrad, high school, industry
- Jobs in FTG
 - Postdoctoral Research Associate in Computer Science
 - Software Engineer
 - Computer Scientist
 - Visit <https://jobs.ornl.gov>
- Contact me vetter@ornl.gov

Final Report on Workshop on Extreme Heterogeneity

1. Maintaining and improving programmer productivity
 - Flexible, expressive, programming models and languages
 - Intelligent, domain-aware compilers and tools
 - Composition of disparate software components
- Managing resources intelligently
 - Automated methods using introspection and machine learning
 - Optimize for performance, energy efficiency, and availability
- Modeling & predicting performance
 - Evaluate impact of potential system designs and application mappings
 - Model-automated optimization of applications
- Enabling reproducible science despite non-determinism & asynchrony
 - Methods for validation on non-deterministic architectures
 - Detection and mitigation of pervasive faults and errors
- Facilitating Data Management, Analytics, and Workflows
 - Mapping of science workflows to heterogeneous hardware and software services
 - Adapting workflows and services to meet facility-level objectives through learning approaches



Bonus Material