Key software challenges in FPGA-enabled heterogeneous platforms

Kazutomo Yoshii <ky@anl.gov>
Mathematics and Computer Science
Argonne National Laboratory

3rd International Workshop for FPGA for HPC (IWFH)
Tokyo, Japan
March 12, 2018

Agenda

- Background and motivation
  - architecture paradigm shift and workload diversity
- High-level synthesis technology
  - pros and cons
- Software challenges
  - abstraction and data movement

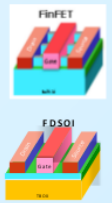
Argonne National Laboratory

Supercomputers

APS accelerator

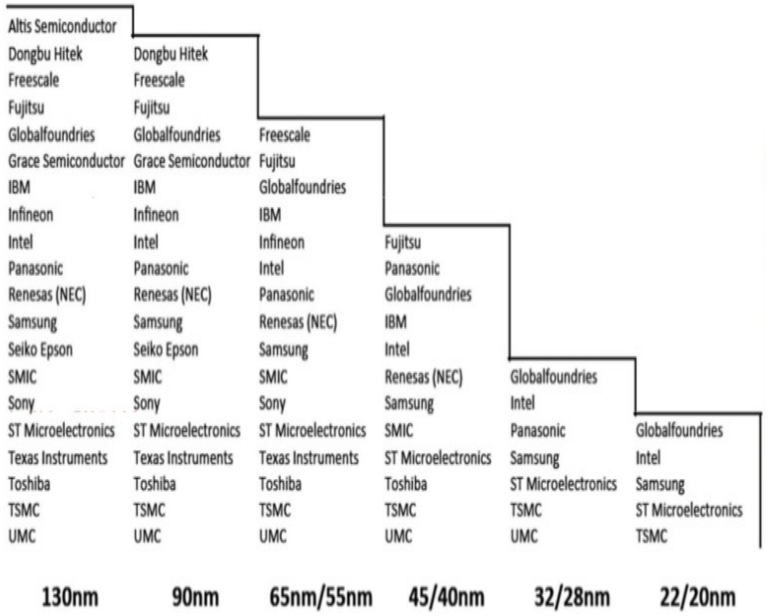
- A DOE national laboratory located near Chicago, Illinois
  - 1,700 acres campus
  - Beautiful 15km trail around the campus!
- Multidisciplinary
  - Hard X-ray, materials, energy storage, nuclear, environmental, high-energy physics, HPC, etc
- A21: the first exascale machine in 2021!

CMOS Scaling Is Coming to An End

Table MM01 - More Moore - Logic Core Device Technology Roadmap							
YEAR OF PRODUCTION	2017	2019	2021	2024	2027	2030	2033
	P54M36	P48M28	P42M24	P36M21	P28M14G1	P26M14G2	P24M14G3
Logic industry "Node Range" Labeling (nm)	"10"	"7"	"5"	"3"	"2.1"	"1.5"	"1.0"
IDM-Foundry node labeling	i10-f7	i7-f5	i5-f3	i3-f2.1	i2.1-f1.5	i1.5-f1.0	i1.0-f0.7
Logic device structure options	finFET FDSOI	finFET LGAA	LGAA VGAA	LGAA VGAA	VGAA M3D	VGAA M3D	VGAA M3D
Logic device mainstream device	finFET	finFET	LGAA	LGAA	VGAA	VGAA	VGAA
Logic device technology naming							

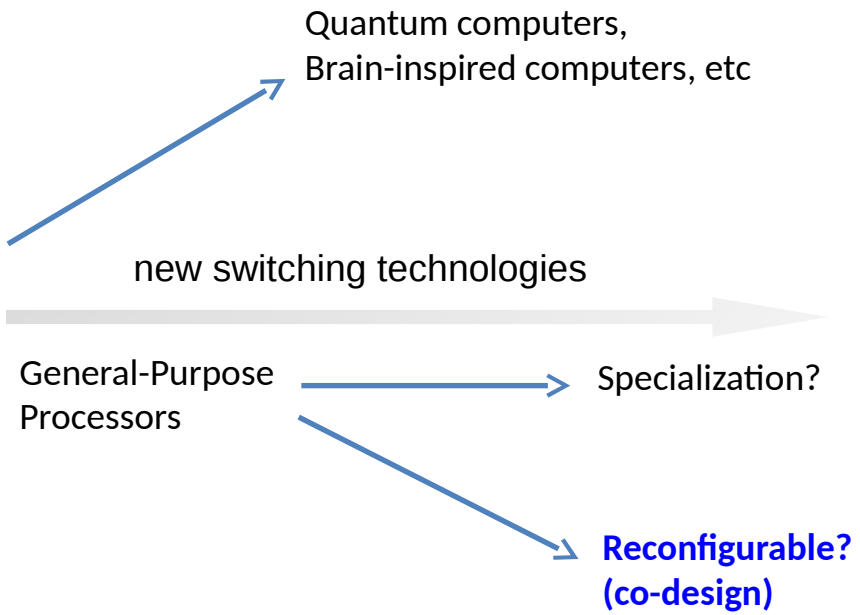
IEEE International Roadmap for Devices and Systems (2017)

- Requires significant investment
  - e.g., Intel spent $5B on 14nm
  - Rock's law: cost of new plant doubles every four years
- Benefits are shrinking
  - thermal, leakage, reliability, etc
- # of manufacturing companies is 20 to 5 in the past 15 years!
- "The number of people predicting the death of Moore's law doubles every two years."



How Can We Survive in The Post-Moore Era?

- Our demands for computational power keeps growing exponentially
  - scientific discoveries depends on computational power
- New types of computers?
  - quantum computers may not be ready in a timely manner
  - different concept, the applicability of classical algorithms is questionable
- Still depend on general-purpose processors
  - Performance is driven by transistor scaling



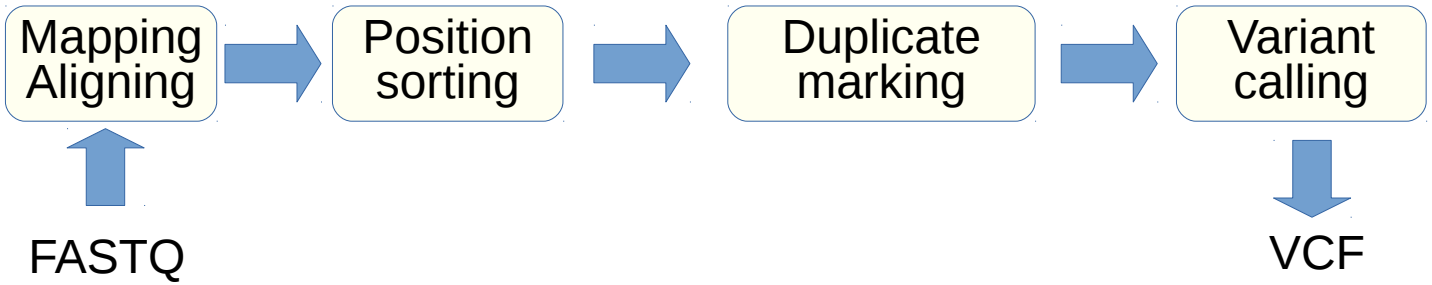
"**Re-form**" LDRD project was funded in 2015

Investigators:
Kazutomo Yoshii, Franck Cappello,
Hal Finkel, Fangfang Xia

Data analytics: genomic analysis

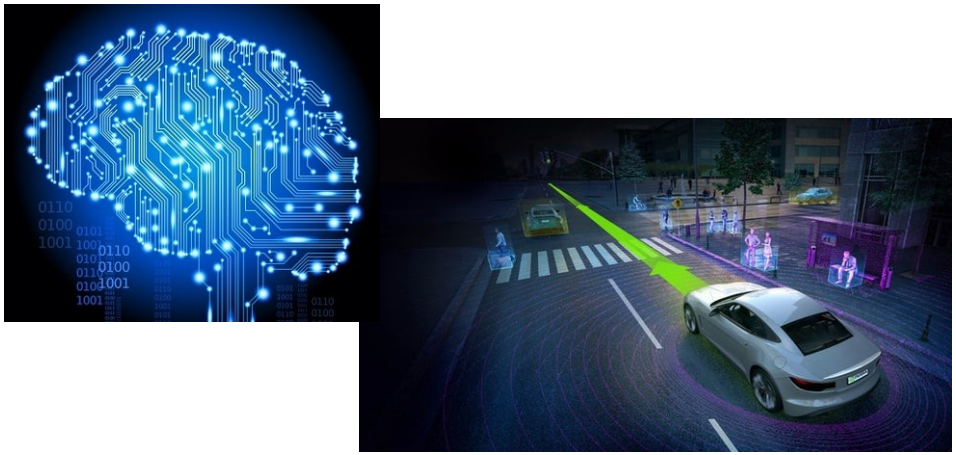
globus genomics

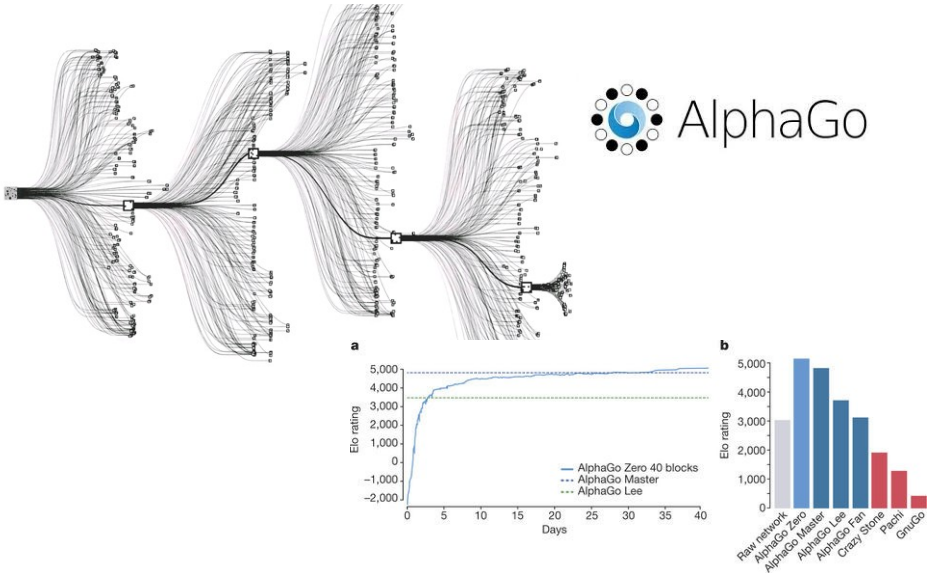
- Complicated pipelines
  - different implementation
  - integer heavy computation on some stages
- Scaling study is still new
  - end up in runtime system development
- Edico Genome currently holds the Guinness world record for fastest time
  - 1,000 FPGA Amazon EC2 F1 instances for 1,000 human genomes



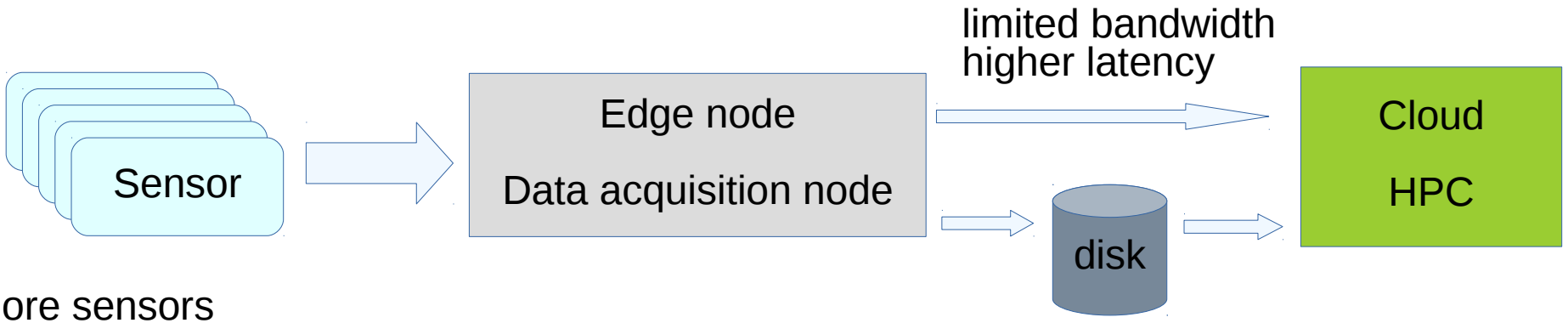
Global Race on AI-Enabled Super Computers

- Machine learning (ML) acceleration is becoming mandatory to future HPC!
- Workload characteristics are different from ordinal numerical computing
  - ML algorithms and implementation techniques keep evolving
    - mixed/reduced precision, stochastic rounding, zero-pruning, etc
    - hard for ASICs
  - Latency sensitive for some application





Edge/HPC integration

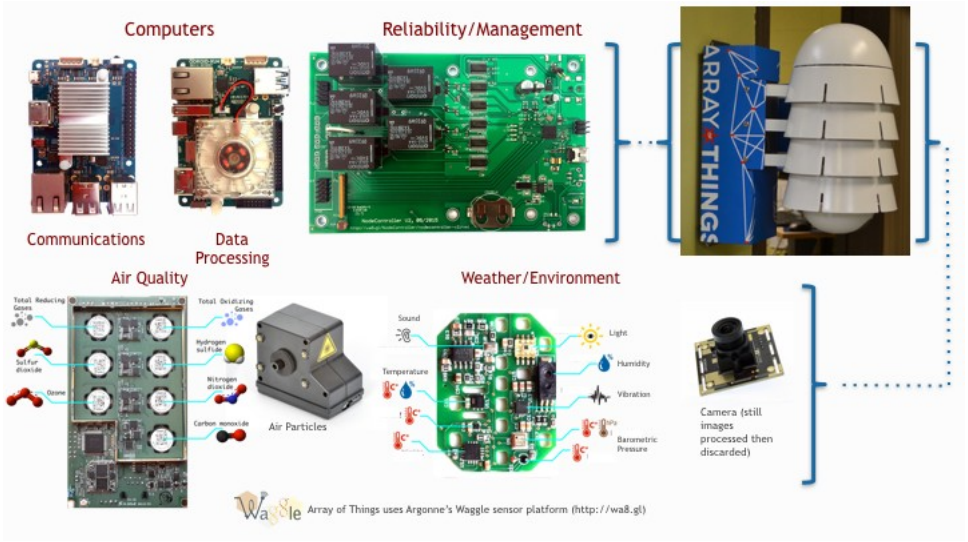


more sensors
data rate is increasing

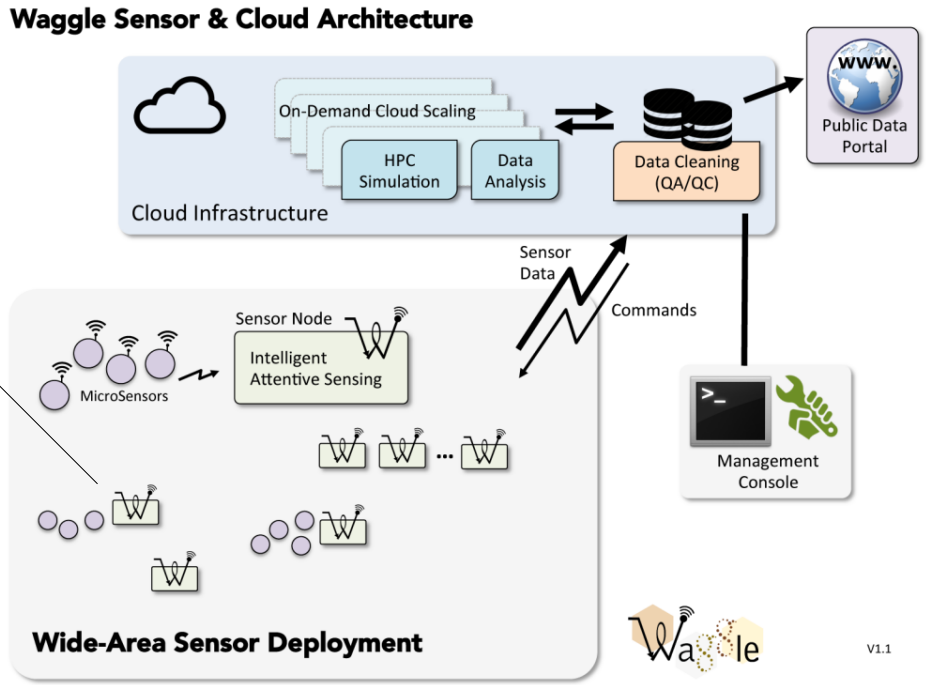
Opportunity here!

old-school to non-FPGA developers

Edge computing



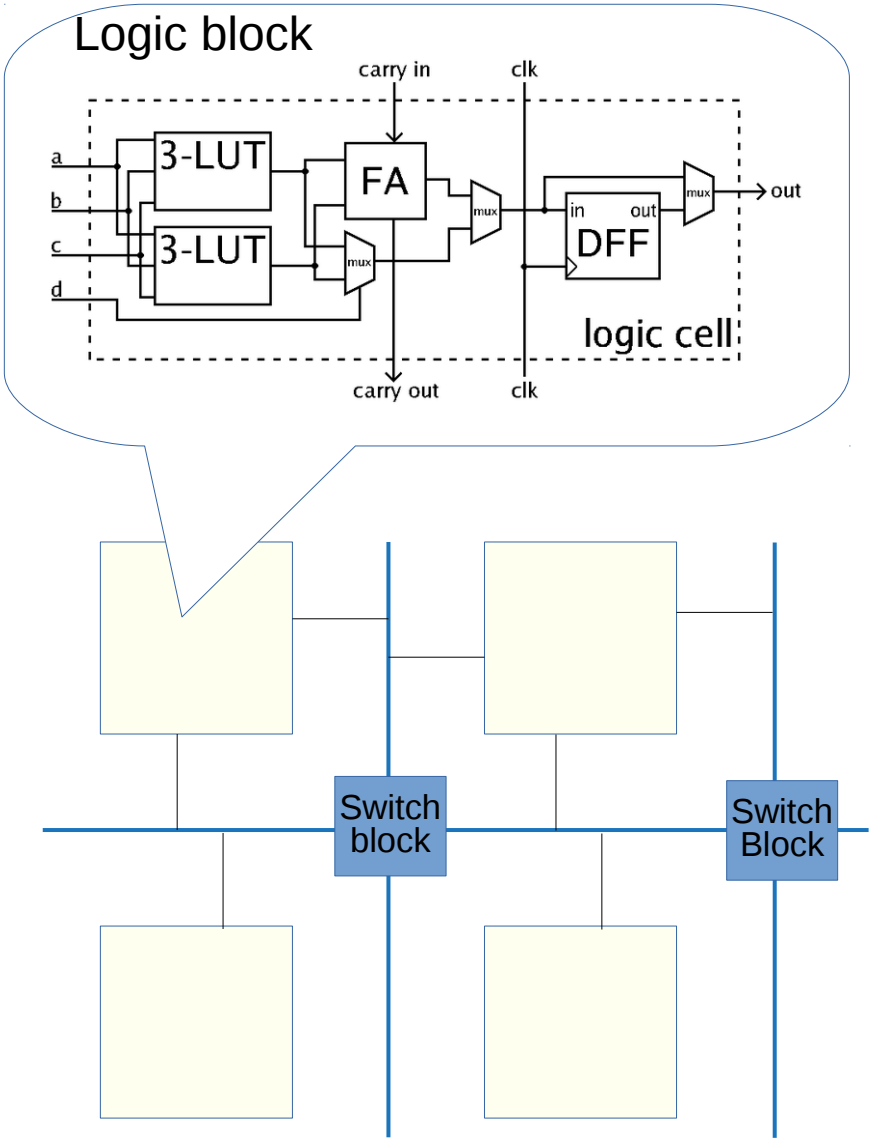
A Waggle platform node



High-level Synthesis Technology

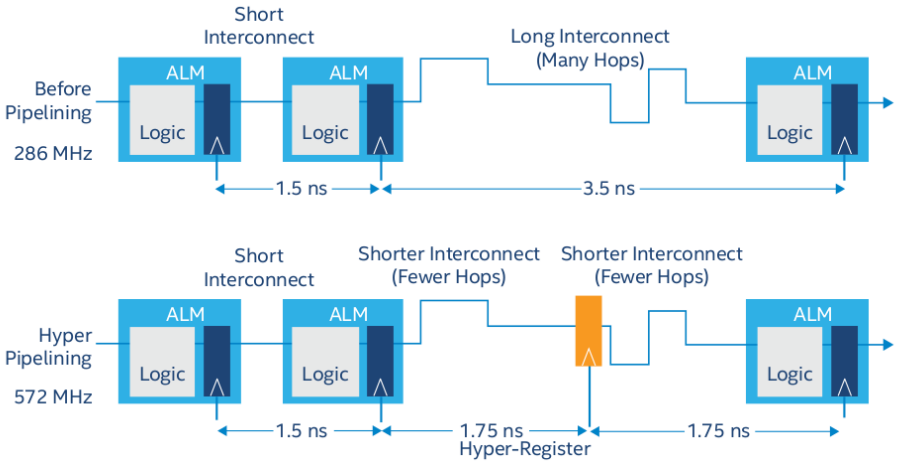
Field Programmable Gate Array (FPGA)

- The first FPGA chip (1985)
  - 64 flip flops, 128 3 lookup tables
- Practical reconfigurable architecture
- Lower non-recurring engineering cost compared to ASICs
  - once a design gets fixed, no one touches
- Application
  - prototype ASICs
  - signal processing
  - data acquisition system



Cutting-edge FPGA technology

- Floating point capability
  - Intel Stratix 10's theoretical peak is 10 Tflops (SP)
- Run faster
  - Technology like Hyperflex helps
  - > 600 MHz is not a dream
- More internal memory
  - up to ~40MB of SRAM (e.g., Xilinx VU37P)
- Off-chip memory improvement
  - HBM2 integration
- High-speed transceivers
  - ~56 Gbps
  - can be used for direct FPGA-FPGA communication
- The advent of FPGA-CPU hybrid platforms
  - ARM-FPGA, Xeon-FPGA, etc
- Embedded-class FPGAs



Can software folks exploit the power of FPGAs?

- Off-the-shell software solutions?
  - OpenCV, tensorflow, cafe, etc

    → very limited now
- Open source FPGA designs?
  - git clone, configure and run easily?

    → not much
- Benchmark results widely available?
  - individual results can be found in scientific publications
  - only a few standard benchmark results
    - e.g., ResNet

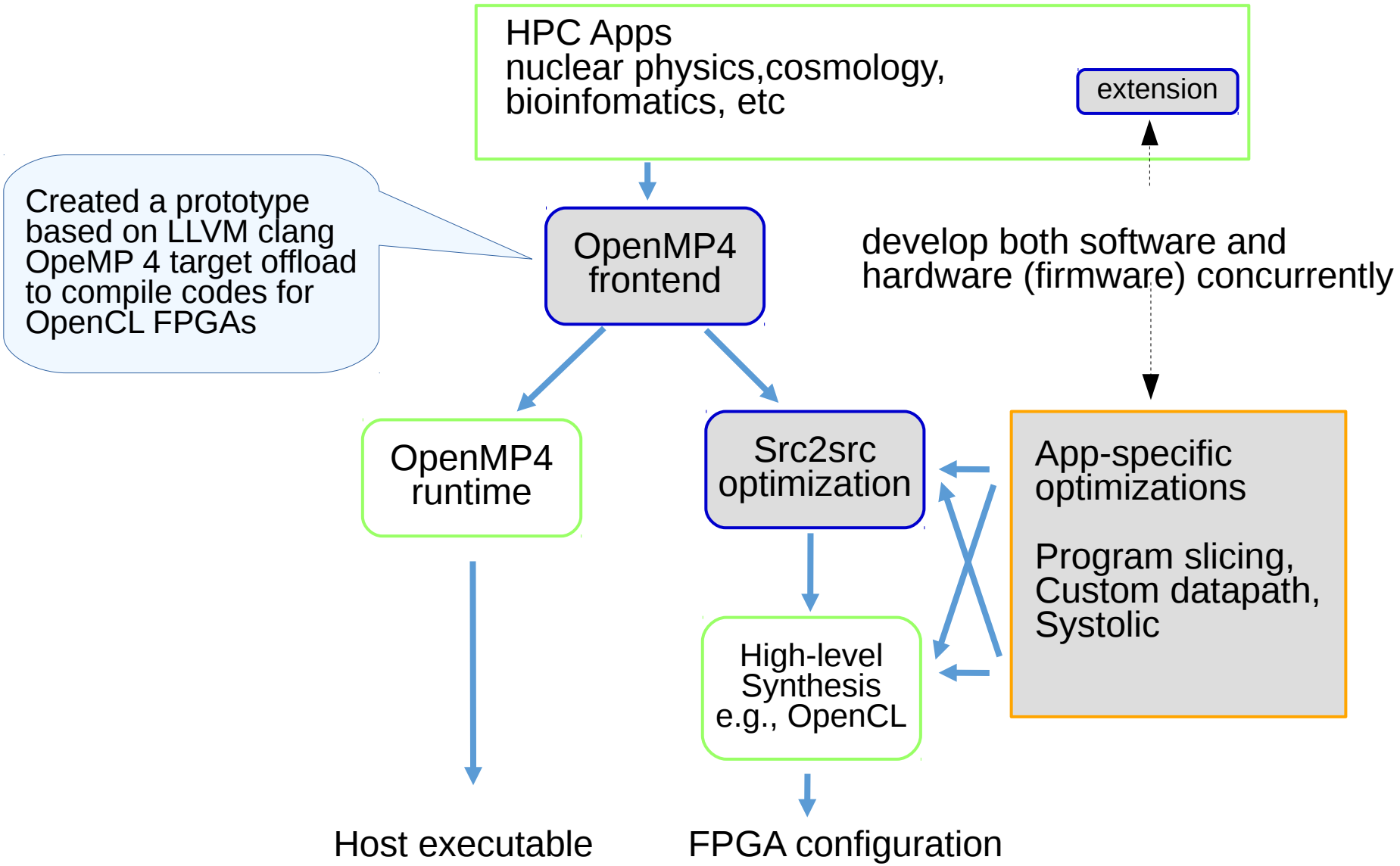
    → same here

Can software folks exploit the power of FPGAs?

- Traditional FPGA design flow is hard for software developers
  - Verilog, VHDL
    - HDL itself may not be a big challenge
    - Too many variations: chip models, boards, tool versions, IP versions, etc
  - longer development, painful verification cycle, lack of abstraction, portability, debugability

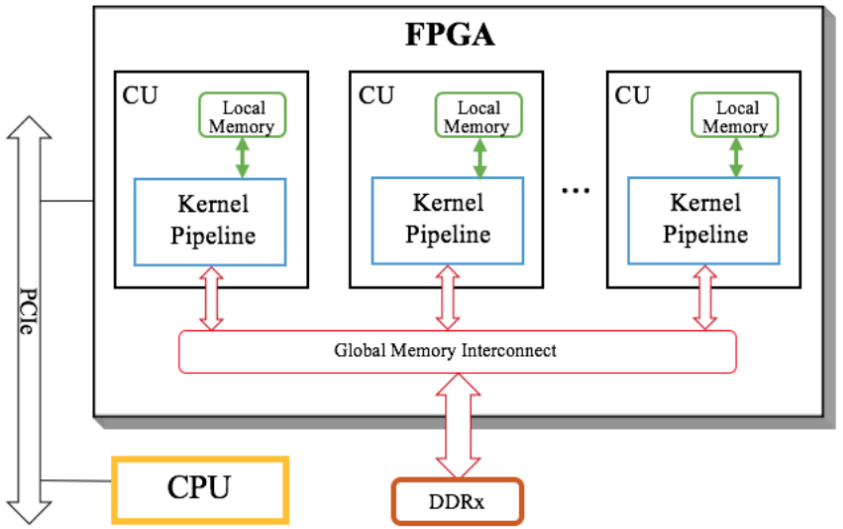
- Can high-level synthesis technology help translate users code into HDL efficiently?
  - following slides
- OS/Runtime questions:
  - How to abstract and manage resources?
  - How to scale?
    - multiple FPGAs, multiple nodes with FPGAs, etc
  - How to virtualize resources?

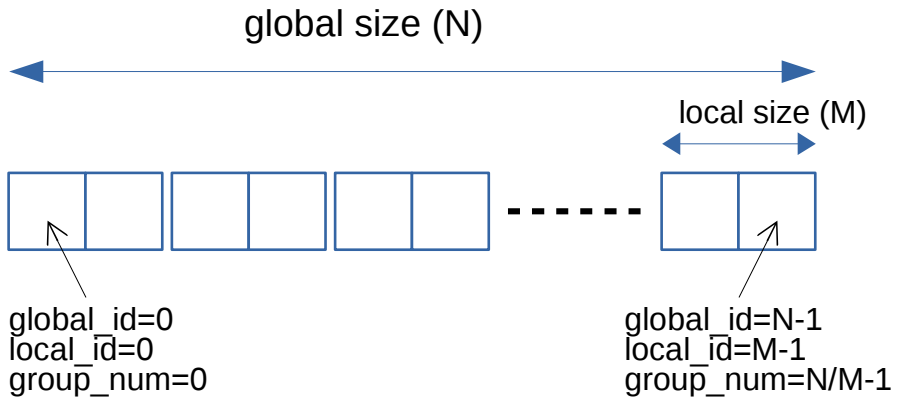
For conventional HPC codes



Evaluation of OpenCL for FPGAs

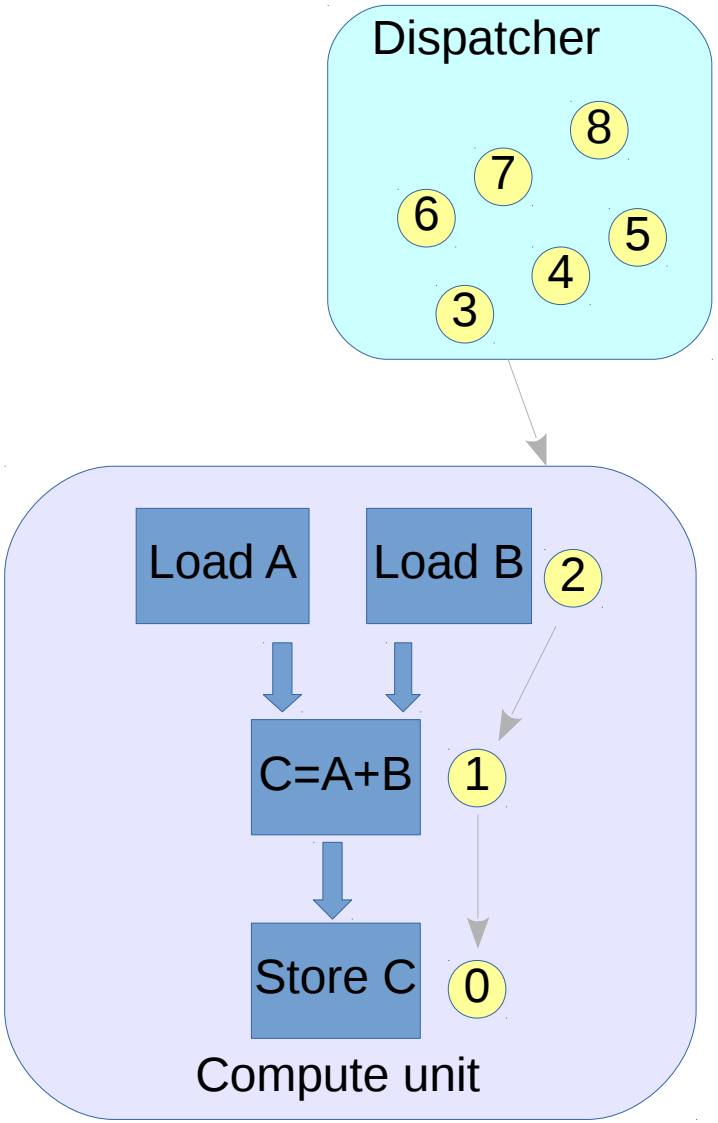
- Most popular high-level synthesis tool
  - Intel FPGA SDK for OpenCL
  - Xilinx SDAccell
- Support heterogeneous platforms
- Support both task and data parallelism (conceptually)
- FPGA specific behavior or extensions
  - # of work-items
  - loop unrolling, SIMDization
  - # of compute units
  - Relaxed floating-point operations
  - control flags for registers, BRAMs





OpenCL data-parallel model: 1D example

Pipeline Parallelism and Dispatcher

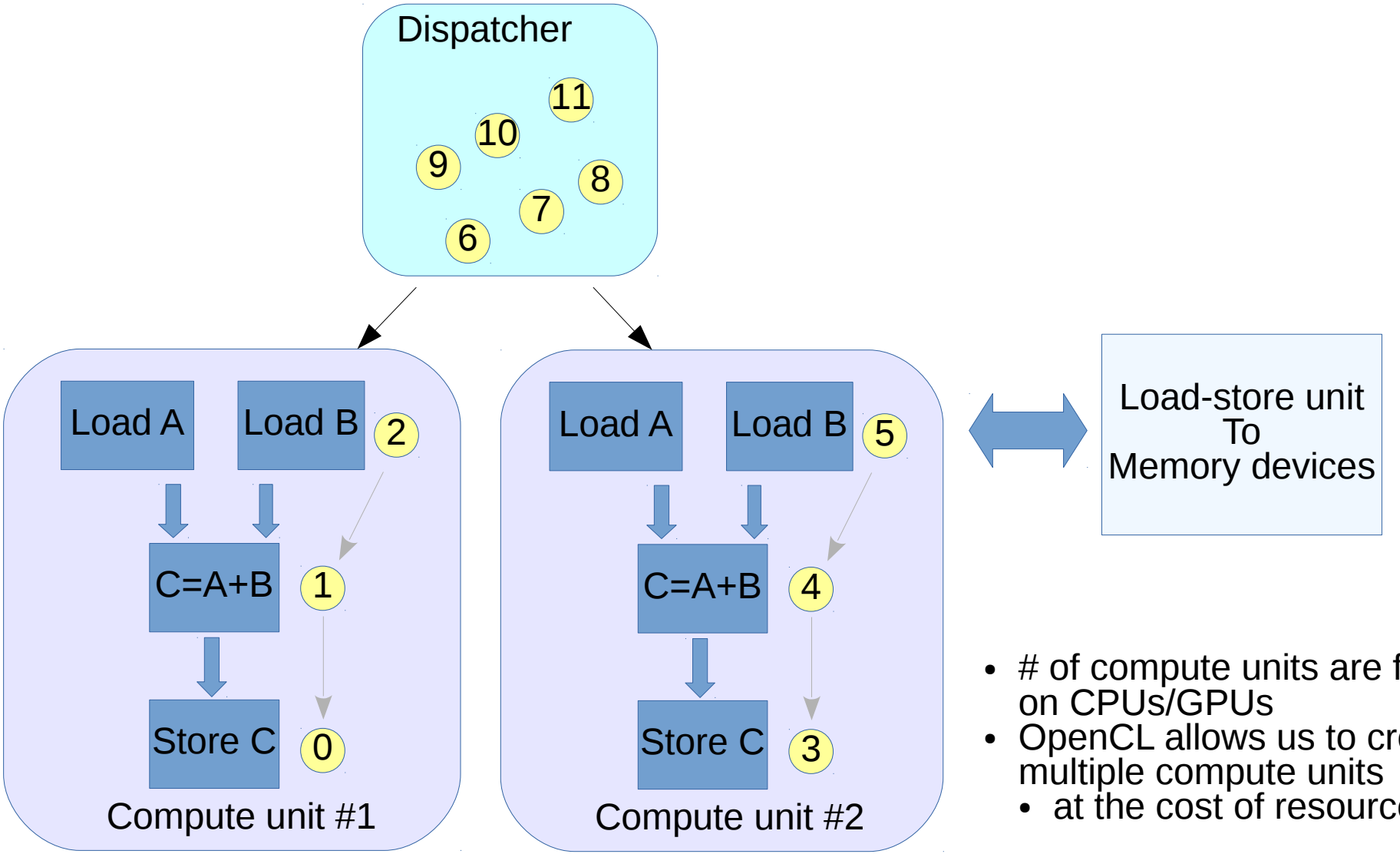



```
__kernel void ndrange_add(A, B, C)
{
    int idx = get_global_id(0);

    C[idx] = A[idx] + B[idx];
}
```

```
__kernel void single_task_add(A, B, C, n)
{
    int i;

    for (i = 0; i < n; i++)
        C[i] = A[i] + B[i];
}
```

- OpenCL creates a single pipeline for both
- Deeper pipeline, higher throughput
- # of compute units is one by default (Intel FPGA SDK for OpenCL)

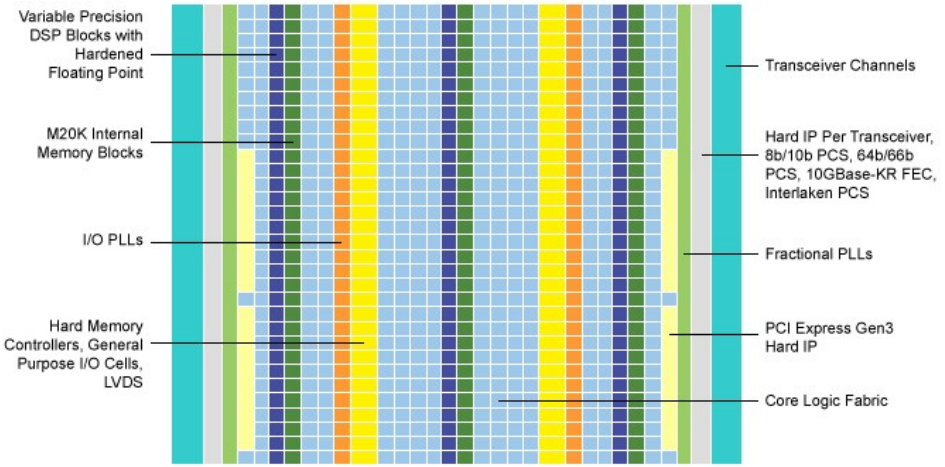
Multiple Compute Units



- # of compute units are fixed on CPUs/GPUs
- OpenCL allows us to create multiple compute units
  - at the cost of resources

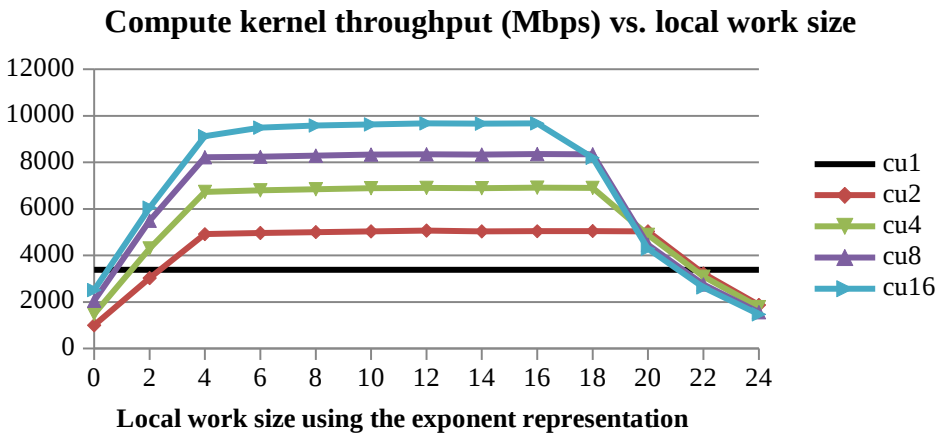
Experiments on Intel Arria10 FPGA

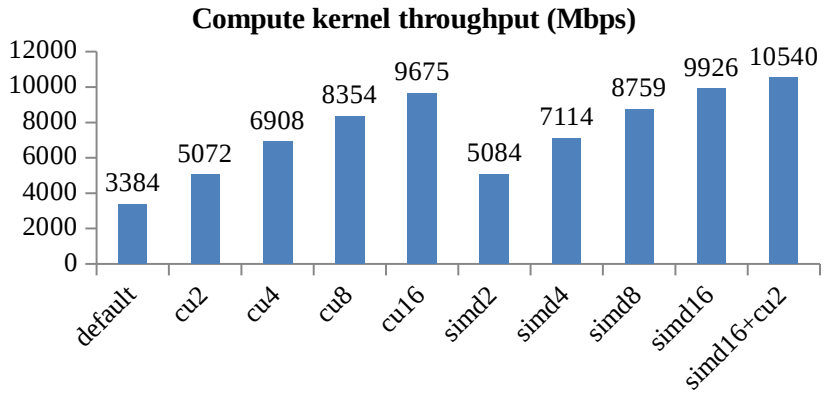
- One of the most popular FPGA chips today
- The first FPGA that supports native IEEE floating-point operations in hardware
- 20nm technology
- Heterogeneous design
- ~1,500 SP hardened FPUs (~3,000 fixed point DSPs)
  - theoretically 1.5 TFlops (SP)
- ~5MB of internal memory
- Hard memory controller
  - supports DDR3/4
  - typical bandwidth 34GB/s (per board)
- PCIe accelerator or CPU-FPGA SoC
- Board vendors generally supports OpenCL



Advance Encryption Standard (AES) Computation

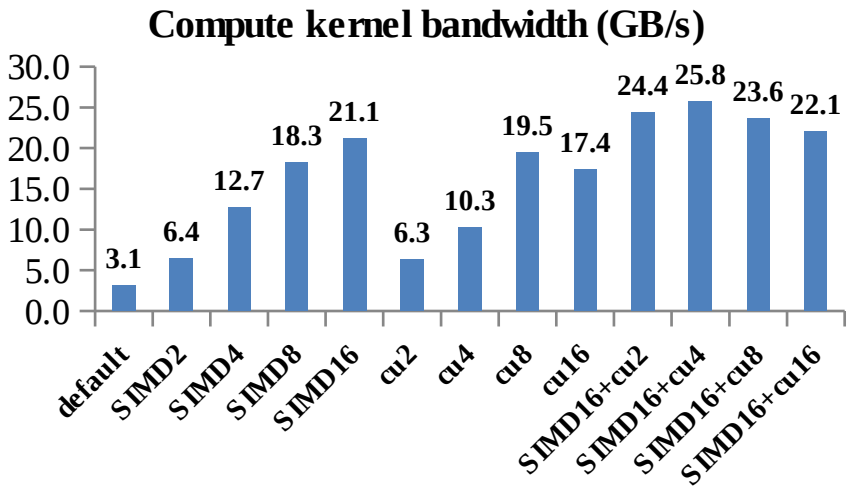
- Ported the AES kernel originally developed by Liu et al. at Virginia Tech to OpenCL
  - AES 256-bit algorithm
  - lots of bit-wise integer operations
  - Input data size: 2GB for 16 blocks
- OpenCL parameters
  - global work size
    - set to the problem size
  - CU: # of computer units
    - one by default
  - local work size
    - how many work-items per CU
  - SIMD: SIMDization
- The performance gain against “default”
  - 3x gain with “simd16+cu2”

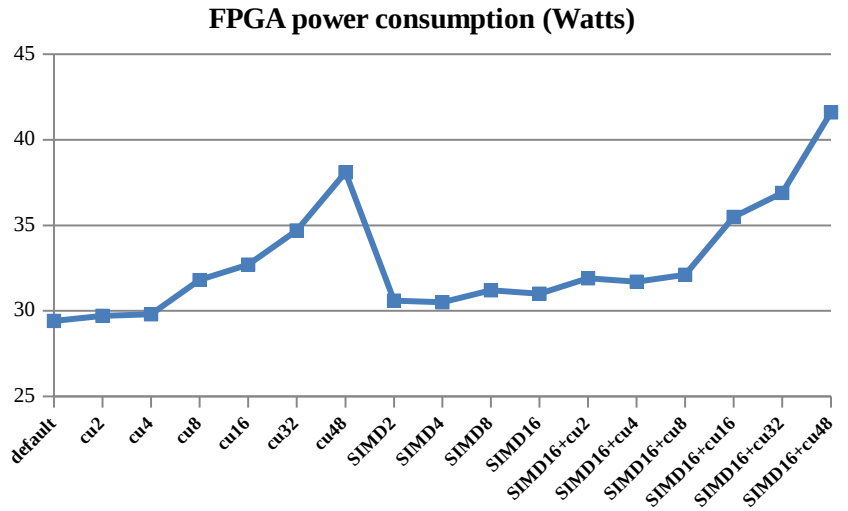




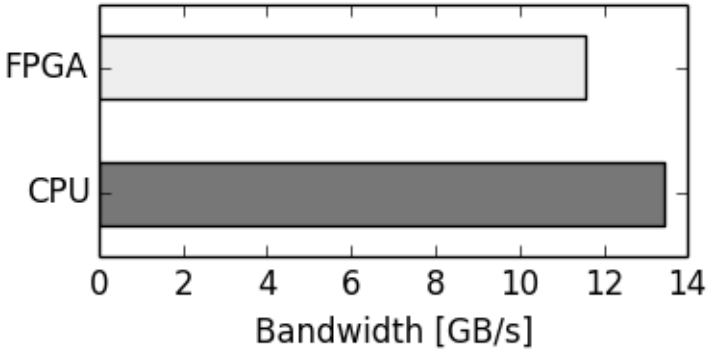
Floating-Point Vector Streaming Add

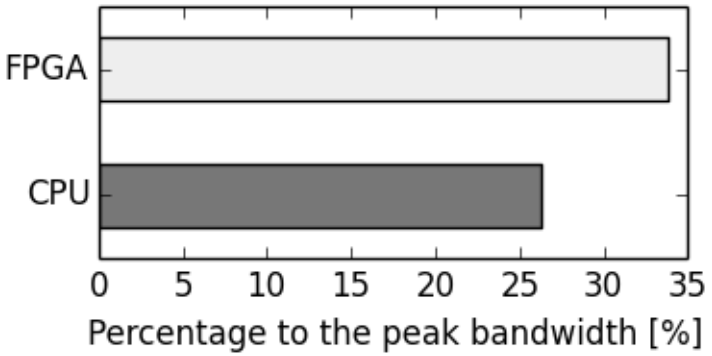
- Memory-bound kernel
- OpenCL parameters
  - CU: # of computer units
  - SIMD: SIMD lane size
  - and combinations
- Observation:
  - Too many compute units cause contentions
    - a diminishing return at cu16
  - The performance gain against “default”
    - 8x gain with “SIMD16+CU4”!

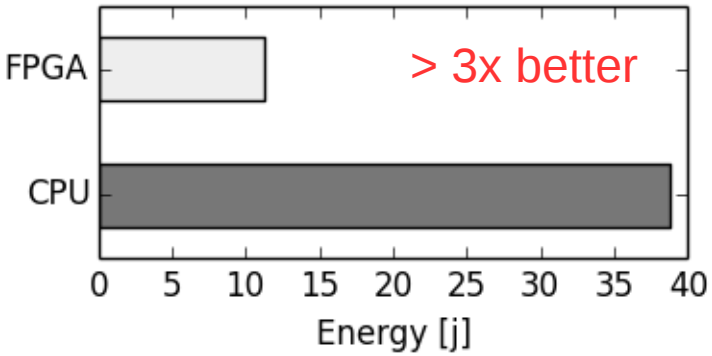




Irregular Memory Access



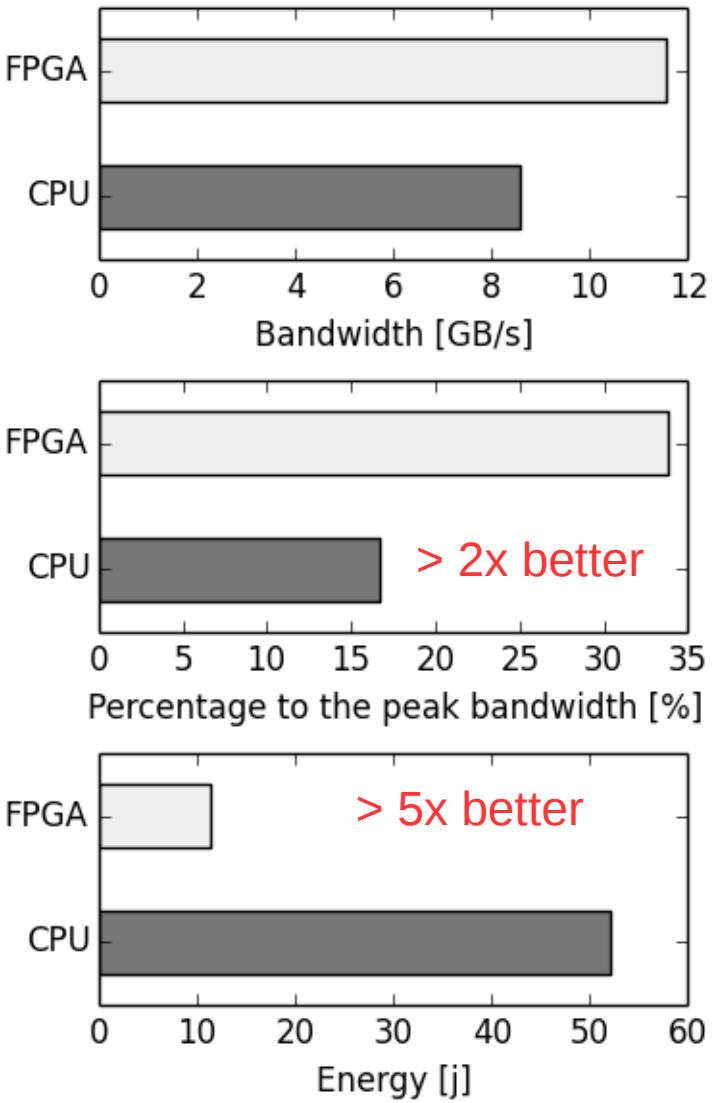





```
for (int i = 0; i < M; i++) {
   double8 tmp;
   index = rand() % len;
   tmp = array[index];
   sum += (tmp.s0 + tmp.s1) / 2.0;
   sum += (tmp.s2 + tmp.s3) / 2.0;
   sum += (tmp.s4 + tmp.s5) / 2.0;
   sum += (tmp.s6 + tmp.s7) / 2.0;
}
```

- # work-units is 256
  - multiple compute units do not improve performance
- CPU: Sandy Bridge with 16 HTs
  - 4ch memory
- FPGA: Arria 10
  - 2ch memory

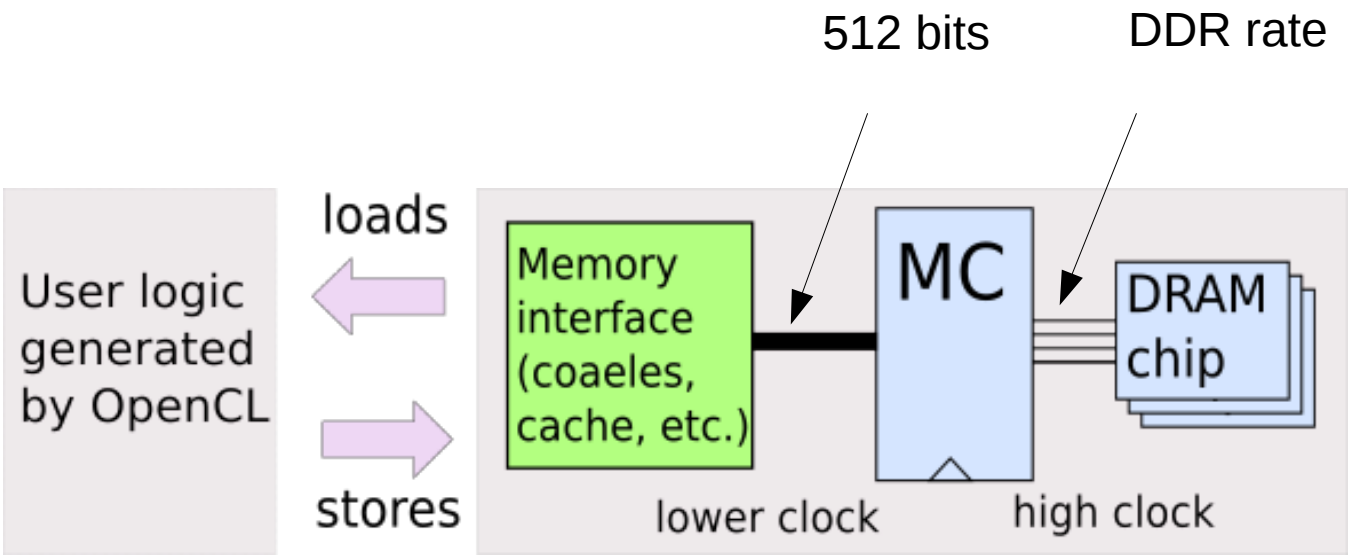
Irregular Memory Access




```
for (int i = 0; i < M; i++) {
   double8 tmp;
   index = rand() % len;
   tmp = array[index];
   sum += (tmp.s0 + tmp.s1) / 2.0;
   sum += (tmp.s2 + tmp.s3) / 2.0;
   sum += (tmp.s4 + tmp.s5) / 2.0;
   sum += (tmp.s6 + tmp.s7) / 2.0;
}
```

- # work-units is 256
  - multiple compute units do not improve performance
- CPU: Sandy Bridge with 16 HTs
  - ~~4ch memory~~ => 2ch memory
- FPGA: Arria 10
  - 2ch memory

Off-chip data movement optimization



Tweaking parameters manually

# of work-items
# of compute units
SIMDization
vector load, store
unrolling
etc

Tools should automate

get rid of such parameters

compiler-level optimizations
e.g., program slicing, custom cache

energy-aware data movement

TRIP: An Ultra-Low Latency, **T**eraOps/s **R**econfigurable **I**nference **P**rocessor for Multi-Layer Perceptrons

- In-memory computation
  - no off-chip memory access
- Demonstrate the current limitation of OpenCL
- OpenCL-Verilog hybrid design
  - MLP engine is written in Verilog
    - use both DSPs and logic blocks for multipliers
    - store weights in BRAM

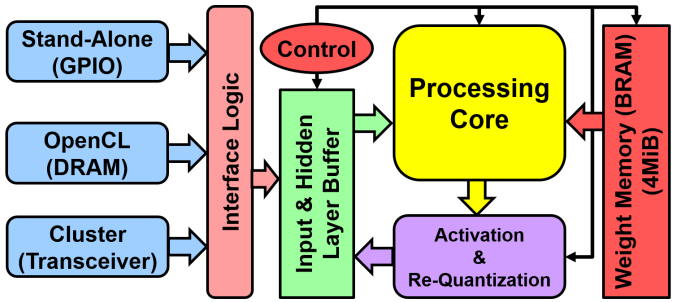
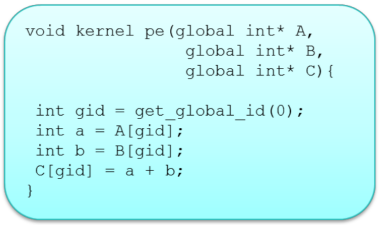


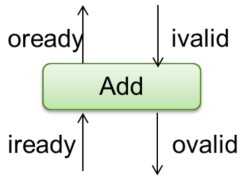
Table 2: ECP-Candle Performance Comparison for Single Input Vector				
Inference Architecture	**M,N**	**Useful Op (%)**	**Performance (TeraOps/s)**	**Speedup**
NVIDIA K80	-	-	0.02	1x
TPU	256,256	79	0.05	2.5x
TRIP Arria 10 CoProc	256,16	91	1.5	75x
TRIP Arria 10 Cluster	256,32	89	3.0	150x
TRIP Stratix 10 CoProc	256,86	88	15.5	775x
TRIP Stratix 10 Cluster	256,102	86	18.0	900x

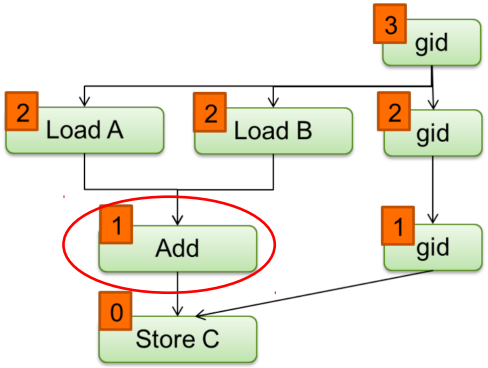
Power	**GPU[1]**	**TPU[1]**	**TRIP**
Idle Power	24 W	38 W	**30 W**
Active Power	99 W	5 W	**2 W**
Total Power	123 W	43 W	**32 W**

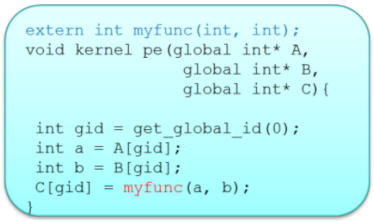
OpenCL-HDL Hybrid Programming Model

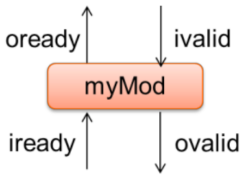
- Intel OpenCL library feature allows us to link OpenCL codes with HDL modules
  - resemble in-line assembly for CPU codes
- The TRIP code needed HDL
  - to implement multipliers using ALMs in addition to DSPs
  - to instantiate BRAMs
- Like HDL development, developers are responsible for verification
  - required to describe latency, stall-free or not, side-effect, etc
- Is this right direction?

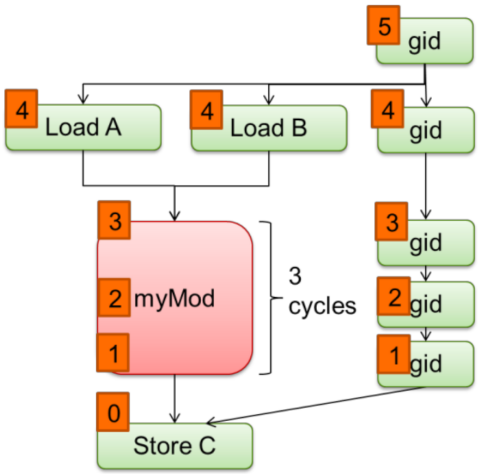












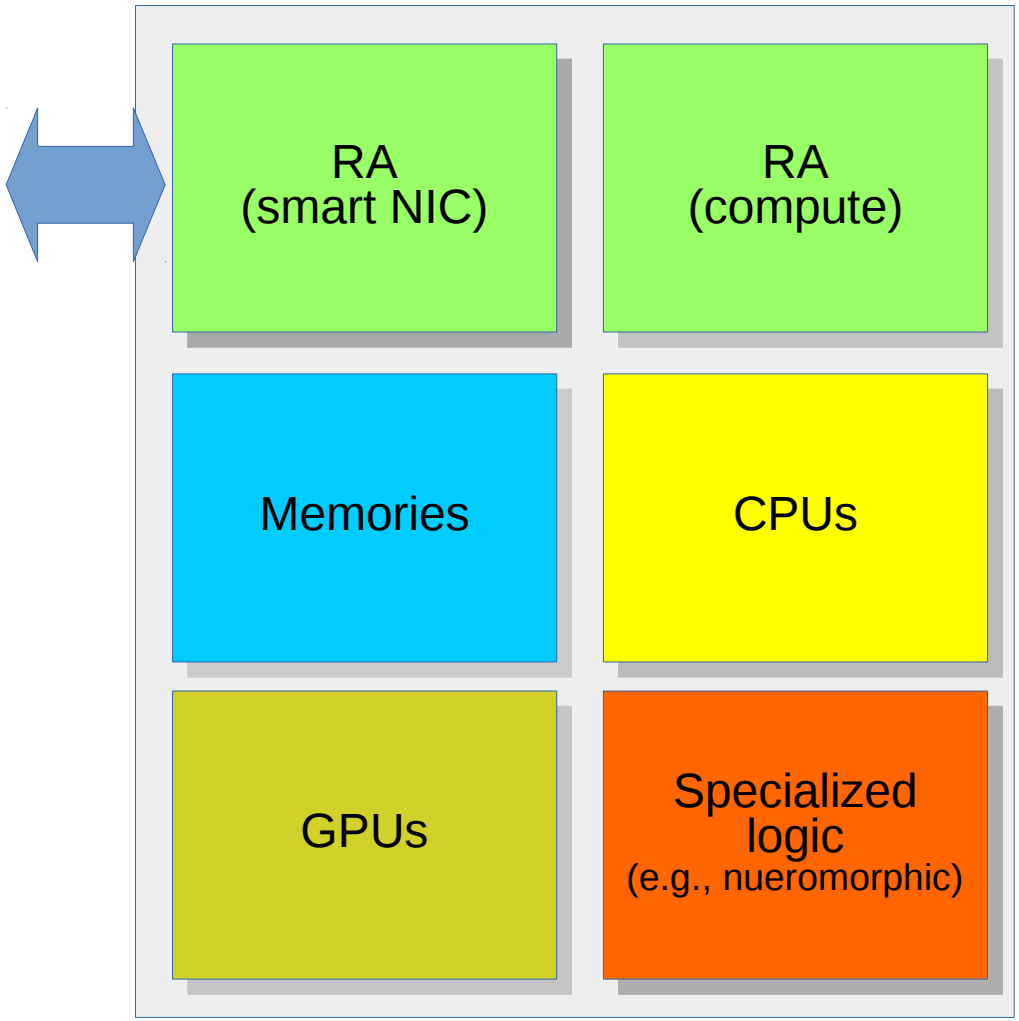
Summary: porting kernels to OpenCL

- Pros
  - CPU emulator is available
  - no timing verification is required
  - medium size kernels such as XSBench, required a week, including debugging
- Cons
  - Performance portability issue
    - FPGA-specific attributes can possibly affect portability
  - OpenCL HDL library offers no portability
  - Longer compilation time affect productivity
- Also need to investigate
  - Compiler-level optimiztion
    - e.g., Falcon computing's Merlin compiler
  - overlay/coarse-grain designs
  - domain specific languages

```
__attribute__((num_compute_units(2)))
__attribute__((num_simd_work_items(4)))
__attribute__((reqd_work_group_size(64,1,1)))
__kernel void foobar(
  __global double *a,
  __attribute__((local_mem_size(4096))) __local float * b,
  __global __attribute__((buffer_location("QDR"))) int *c) {

  int __attribute__((memory,
      numbanks(1),
      bankwidth(128),
      doublepump,
      numwriteports(1),
      numreadports(4)) c[32]; // too much hardware detail

  ...
  ...
}
```


Software challenges

Heterogeneous node designs

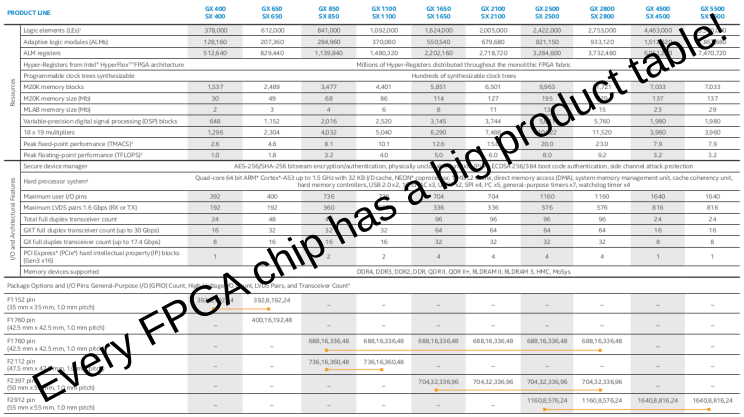


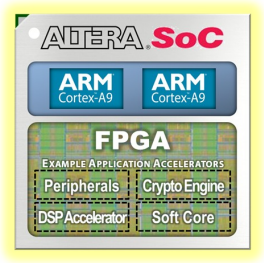
- Gain performance and energy efficiency
- Heterogeneous designs are already adopted by smartphones
  - iPhone7 has an FPGA chip
- Accelerating:
  - computation. e.g., simulation, data, learning
  - OS and system-level tasks
  - network/storage
    - compression, streaming computing, etc

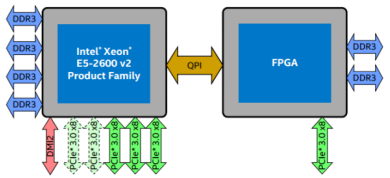
Software lag behind hardware!!!

Lack of abstraction

- So many FPGA chips, different boards/platforms
- No compatibility, even source-level
  - longer compilation time
  - so many different Xeon chips, too, but they offer binary compatibility and shorter compilation time
- HLS can abstract FPGA resources to some degree
- Need to abstract off-chip memory, I/O, debugging APIs, etc

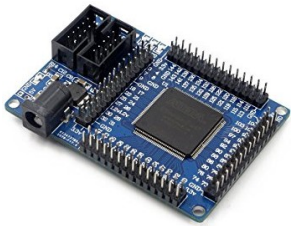






Tightly-coupled w/ beefy CPUs

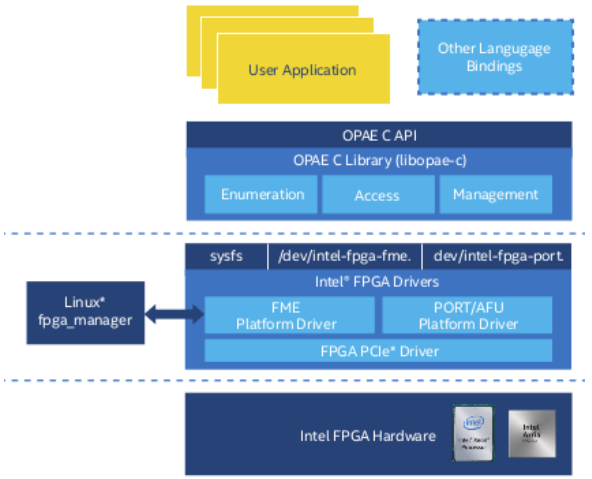
PCIe



Standalone

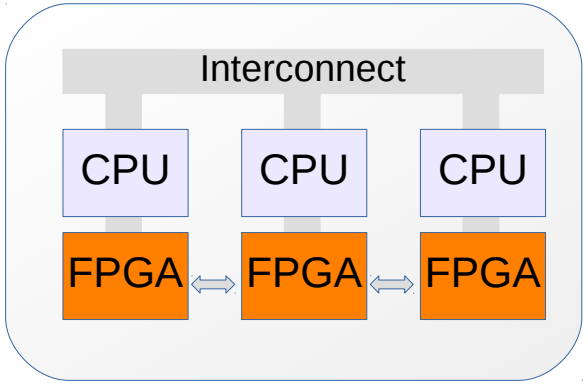
Existing abstraction layers for FPGAs

- OpenCL BSPs
  - provide low-level software APIs
  - limit board choices, less extensible
    - or build a custom BSP
  - OpenCL features may be an overkill
- Intel Open Programmable Acceleration Engine (OPAE)
  - abstract accelerator such as FPGA
  - consists of kernel drivers, userspace libraries and tools (e.g., discover, reconfigure)
  - Supported platforms ?
- AWS EC2 FPGA hardware and software development kits
  - FPGA shells and software APIs
  - https://github.com/aws/aws-fpga.git



- Questions
  - support our edge-to-HPC needs?
  - support various programming models?
    - offload, streaming, pure dataflow, hybrid dataflow, etc

Identifying requirements for abstract layers (on-going)



User app

Middle Layer (task, dataflow, etc)

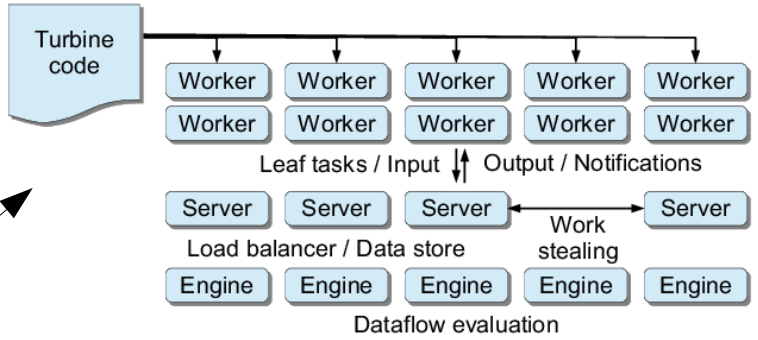
Userspace APIs and runtime

User code or library or overlay

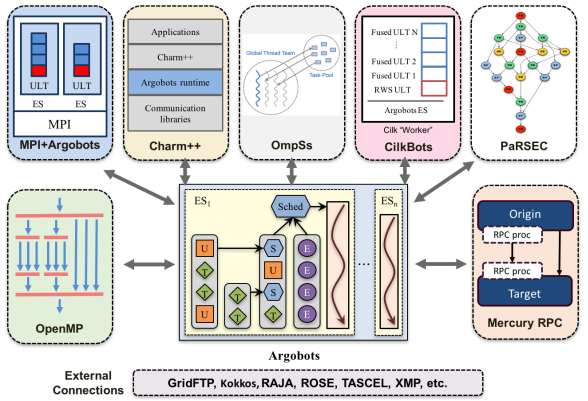
FPGA shells

Kernel drivers

reconfigure
diagnostic
data stream, copy
....

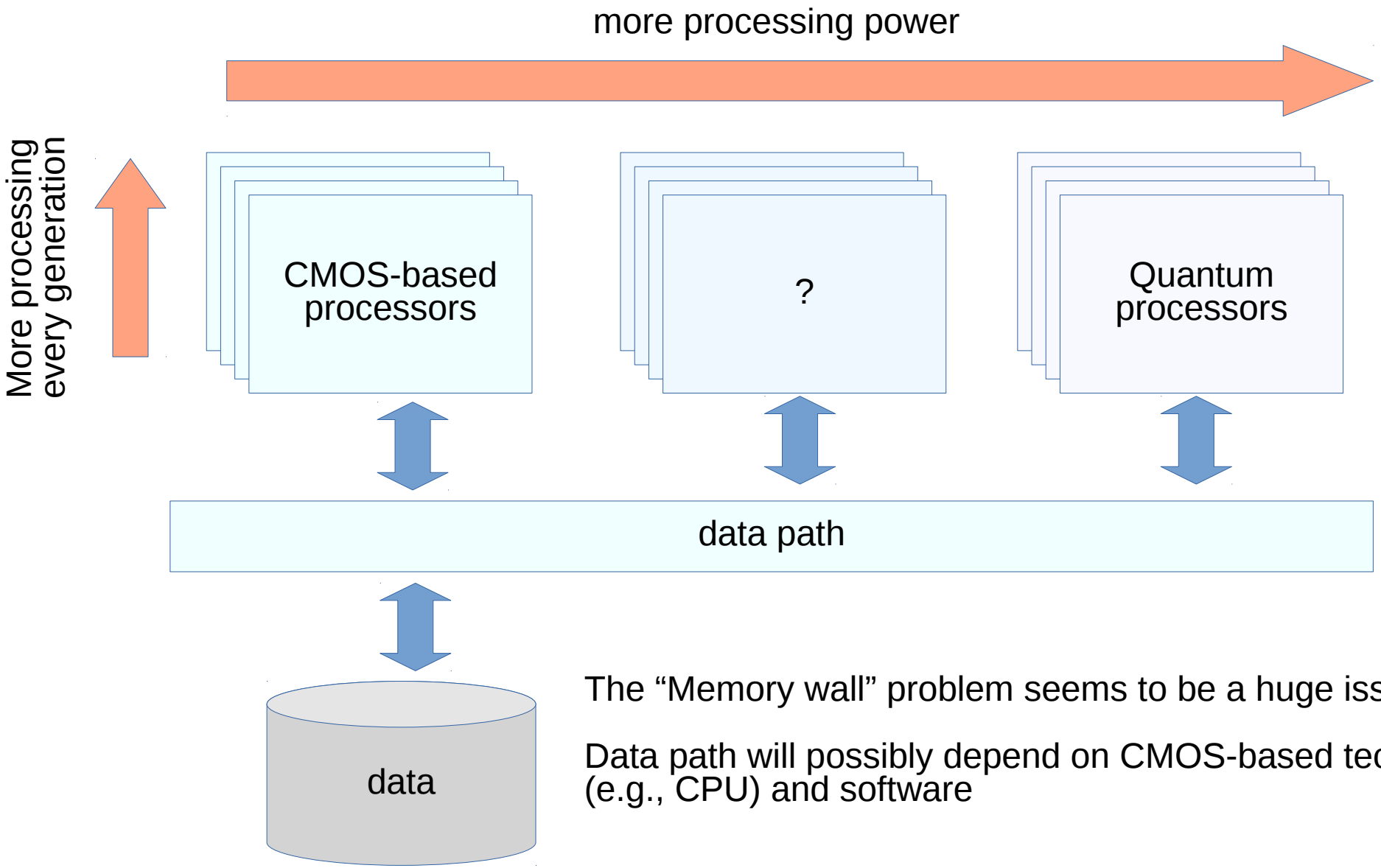


Swift/T



Argobots

Extreme heterogeneity

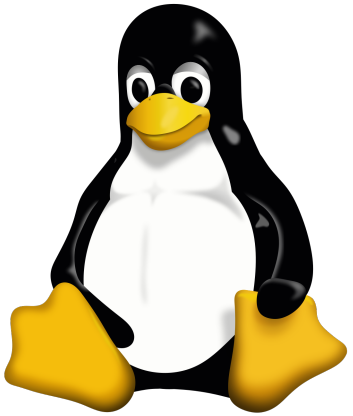


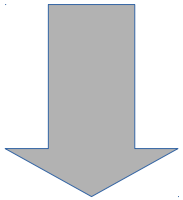
The “Memory wall” problem seems to be a huge issue

Data path will possibly depend on CMOS-based technology (e.g., CPU) and software

Conclusion

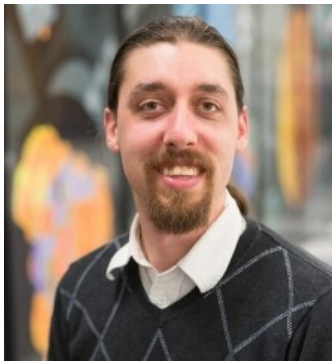
- Identify right-level of abstraction
  - learn from Andoid?
- Leverage dataflow, task-based model
  - to hide details
  - optimize data movement
    - deal with deep hierarchy
    - energy-aware manner
- Study OS/R-level concepts
  - flexibility and performance





People Involved in The “Re-form” Project

Franck Cappello
(MCS)



Hal Finkel
(ALCF)

Fangfang Xia
(CELS)

Kazutomo Yoshii
(MCS)

Zheming Jin (ALCF)

[Students]

Yingyi Luo (Northwestern University)

Ahmed Sanaullah (Boston University)

Chen Yang (Boston University)