

Reconfigurable Computing: an Ingredient of Post-Moore Scientific Computing?

Franck Cappello (ANL)

With contributions from:

Kazutomo Yoshii, Hal Finkel, Fangfang Xia (ANL)

Yingyi Luo (NU)

ATPESC

Argonne Training Program on Extreme-Scale Computing

Summary

- End of Moore's law
- 3 trends toward tailored hardware
- Current FPGA applicability
- Reality check



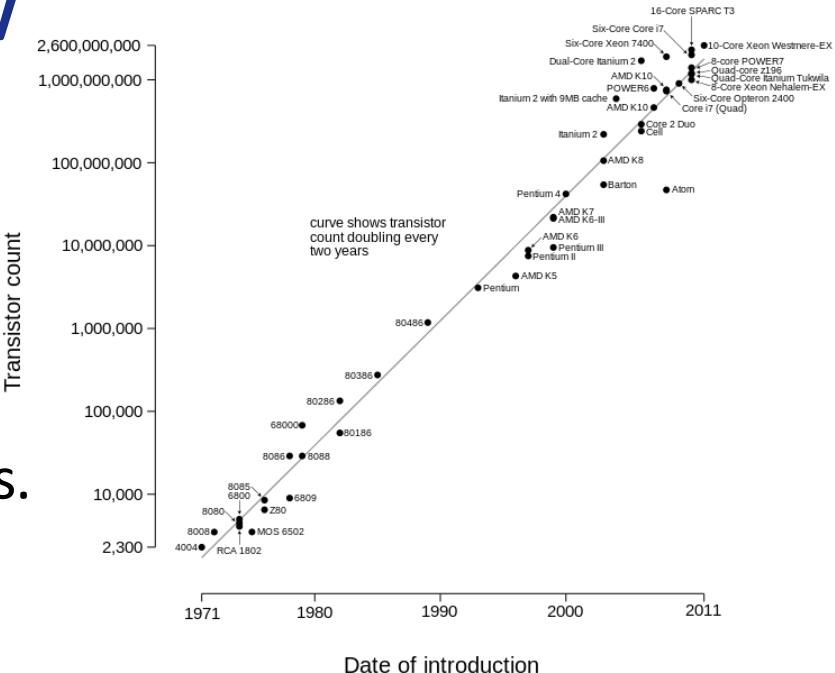
End of Moore's law

The Original Moore's Law:

Usually cast as doubles the number of transistors per chip, at constant cost every 18-24 months.

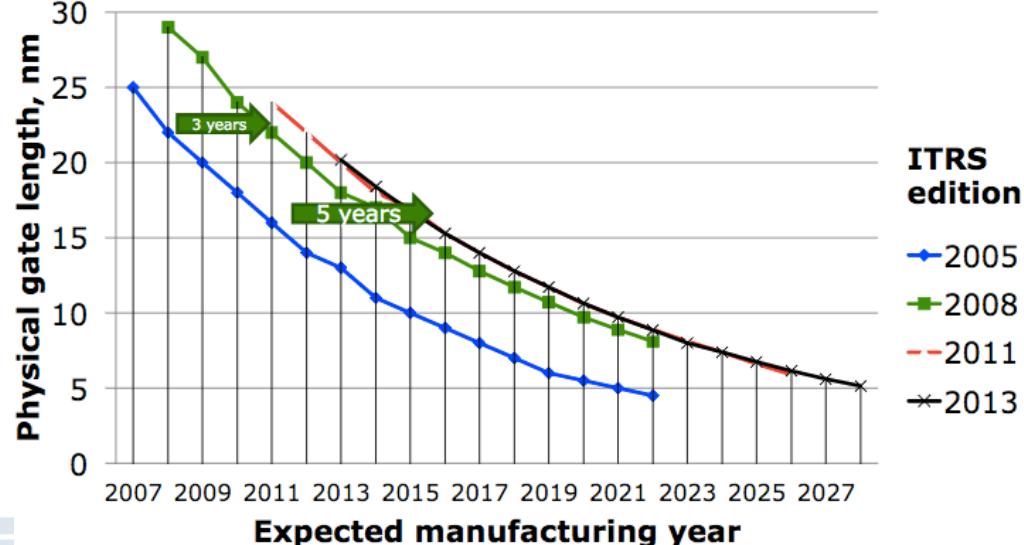
→ Observation made in 1965

Microprocessor Transistor Counts 1971-2011 & Moore's Law



Exponential improvement very difficult to maintain.

ITRS projections for gate lengths (nm) for 2005, 2008 and 2011
→ A shift of 5 years in 4 years!
→ 5nm may never happen



End of Moore's law

What does this mean for CMOS based processors?

→ Density will be bounded (# of transistors per surface unit stays the same)

Exploit the 3rd dimension. Yes but:

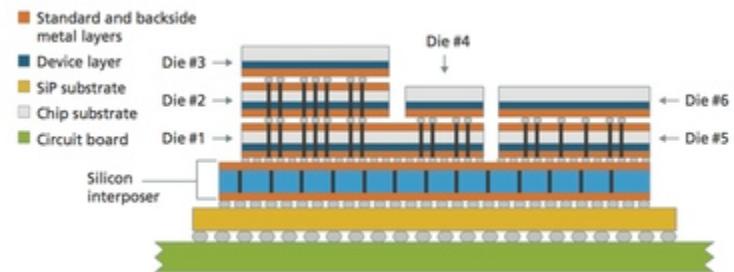
→ 1) there is only 1 dimension to exploit to achieve improvement (exponential in time?)

→ 2) there will be a limit (power, heat, testing, cost?)

“No tree grows to the sky”

→ What happens if #transistors, Frequency, and Power are bounded?

→ How to still improve performance?



End of Moore's law

Post Moore law
Technology + Architecture

New technologies:
Carbon nano tubes,
nanoscale vacuum
channel transistor, TFET,
spin devices, Josephson junction,
Single atom transistor:
→Unlikely to keep-up with
Exponential performance growth

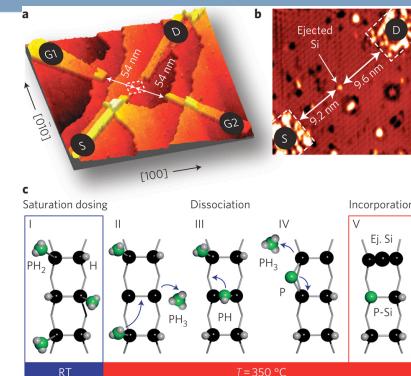
Slowly
Improving
CMOS and
new
technologies

-Process variation
(gate length, wire width, etc.)
-Leakage current



Reconfigurable
(co-design)

Specialized
Architectures
(co-design)



A single-atom transistor Martin Fuechsle, Jill A. Miwa, Sudhasattha Mahapatra, Hoon Ryu, Sunhee Lee, Oliver Warschkow, Lloyd C. L. Hollenberg, Gerhard Klimeck Michelle Y. Simmons Nature Nanotechnology 7, 242–246 (2012)



Neurosynaptic
Graph processor
Anton (FFT, etc.)

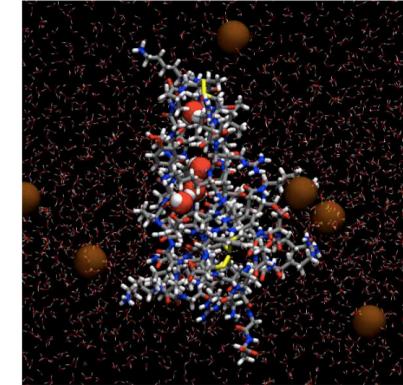
Specific systems: Anton 2 machine



Presented at HotChip 2014 (August)

Specific application: Molecular Dynamics

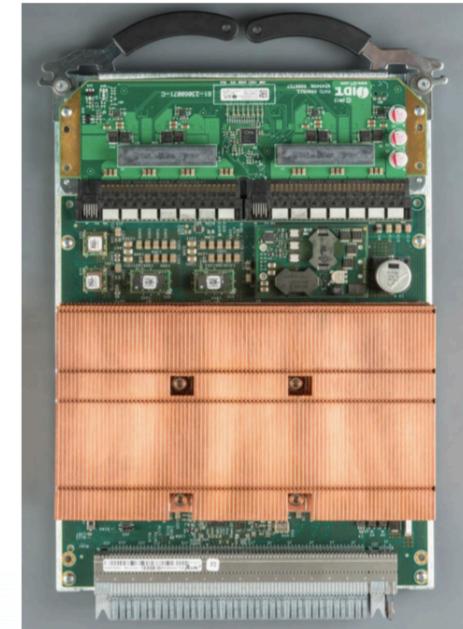
Simulation of the motions of all atoms in a biochemical systems
 $10^5 - 10^6$ atoms
Time step: 2fs, 10^{12} time steps



- A specific Chip (ASICs:
Application-specific integrated circuit)
- 66 CPU cores
- 1,650 MHz
- 12.7 TFXOPS (32 bits fixed point)
- 250 W
- 40nm CMOS technology.
- Same year: Intel Broadwell in 14nm
- Tesla K80 (8.73 Teraflops single-precision, 28nm), 300W



→ Large heat sink



Specific systems

[Jun Makino](#)

Tokyo U. in front
of the Grape4
Gravity engine



Interesting but:

- Long to develop
- Not the latest technology (at least in the past)
- Need to design for fixed algorithm or small family of algorithms
- Cannot run efficiently on a large variety of applications
- Very expensive (considering the diversity of applications)

What happen is #transistor, frequency and power are bounded?

- improvement come only from new algorithms
- Need to redesign and build an new ASIC at each significant improvement



Four trends push toward reconfigurable hardware for scientific computing

- 1) FPGA are getting competitive on Floating Point performance and connectivity (WRT GPUs, CPUs)
- 2) Data becomes the main performance constraint
- 3) Reduce power&size of operators to increase performance
- 4) It is already adopted in data centers



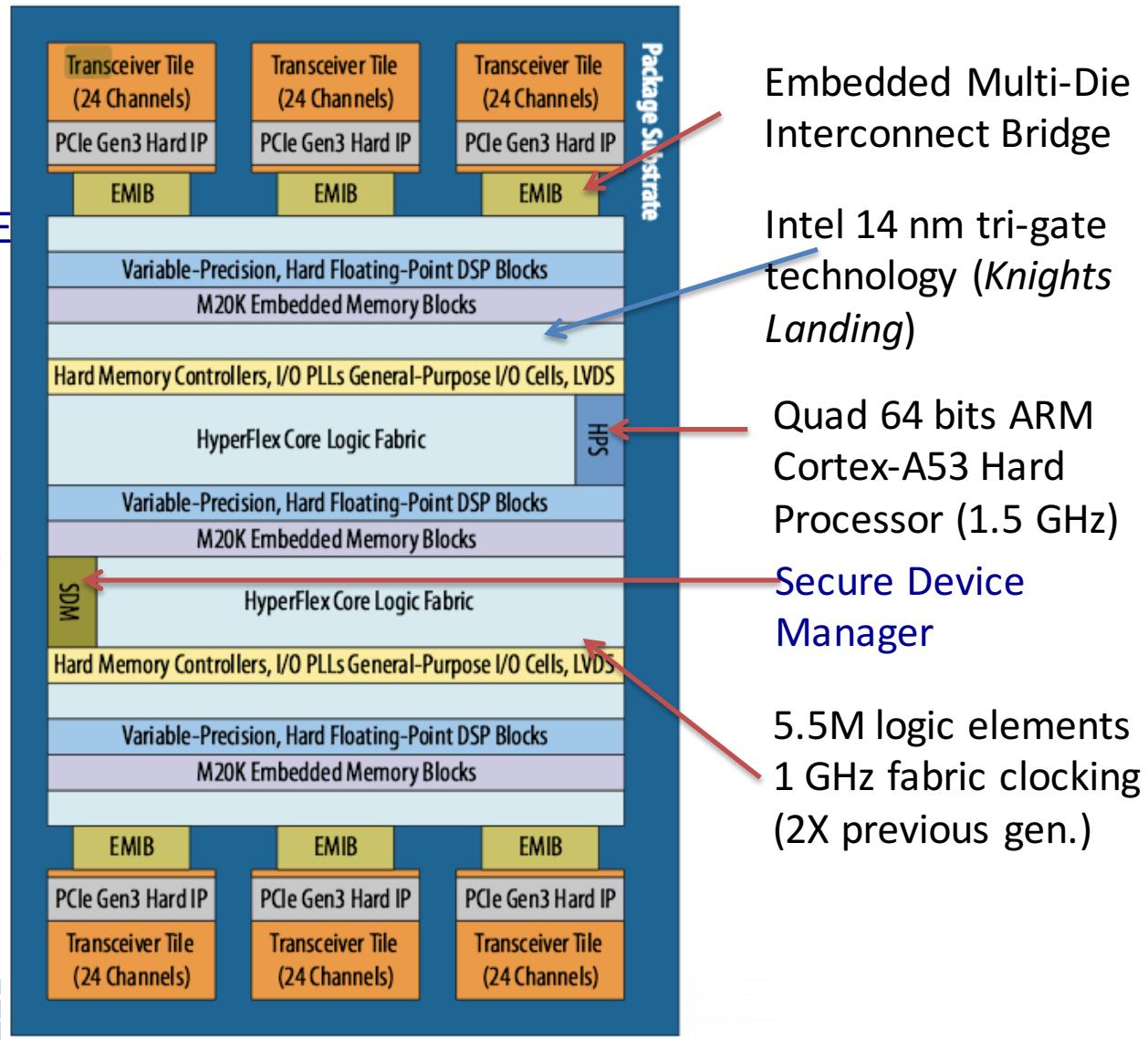
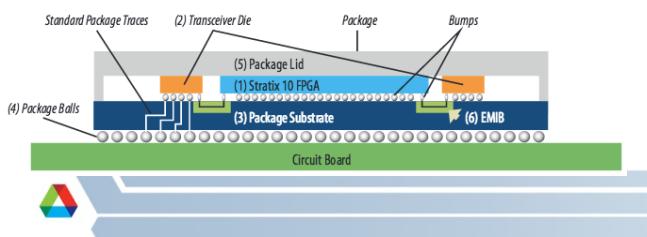
1) New FPGA are getting competitive (against GPUs and CPUs)

Example:

Stratix 10 SoC:

- 10 TFLOPs (single-precision IEEE 754 DSPs)
- ~100 GFlops/Watt (SP)
- 64 bit quad-core ARM (1.5 GHz)
- 1 Tbps bandwidth (Hybrid Memory Cube)
- Up to 144 30 Gbps transceivers
- 5.5M logic elements (FPGA)
- 1 GHz fabric clocking (2X previous gen.)

Multi-die on substrate



1) New FPGA are getting competitive (against GPUs and CPUs)

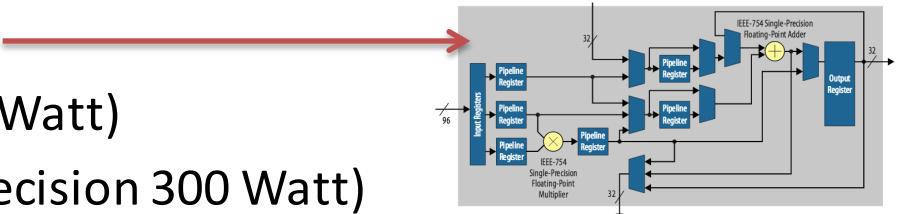
What really could be game changer:

- Variable Precision DSP Blocks

10 TFLOPS peak, single precision, 125Watt)

(Nvidia Tesla K80: 8.7 Tflops single precision 300 Watt)

((DOE SUMMIT « fat » nodes will provide 40 TFLOPS peak))

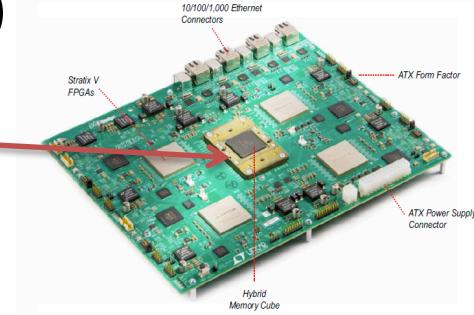


- Hybrid Memory Cube.

Up to 64 channels, each with data rates up to 15 Gbps

→ up to 1 Tbps aggregated bandwidth/direction

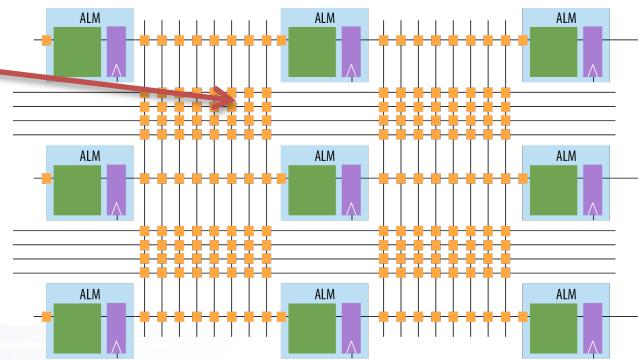
Sorry: no data on data movement power consumption



- In core fabric bypassable registers

A Register associated with each routing segment

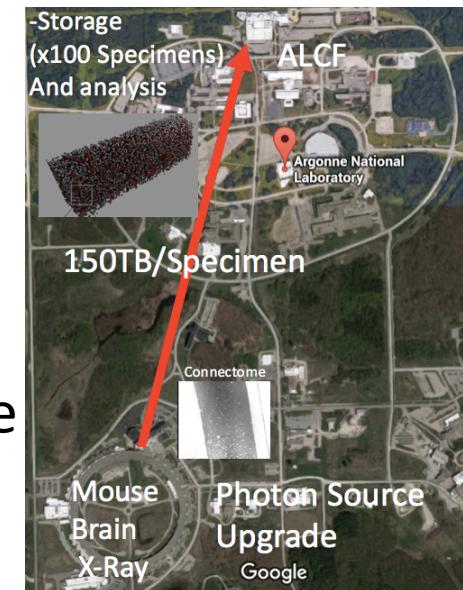
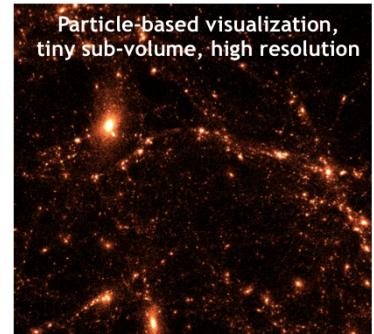
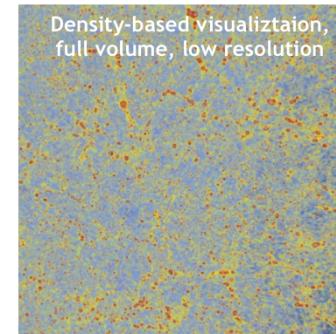
(clock depends on the longest section between flipflops.)



2) Data as the main constraint

Planned scientific simulations and experiments generate many more data than can be stored or communicated.

- Examples:
 - Cosmology Simulation (HACC):
 - A total of **>20PB** of data simulating trillion of particles
 - Petascale systems FS ~20PB
→ data reduction is needed
→ currently drop 9 snapshots over 10
 - APS-U (next-generation APS project)
Brain Initiatives: high-throughput x-ray tomography of whole mouse brains.
 - 160TB of storage per specimen
 - hundreds of specimens → **100PB** of storage
 - APS-U to ALCF:20Gb/s link
→ >1 year to transfer the data



2) Data as the main perf. constraint

FPGA are pretty good on data compression

- M. S. Abdelfattah, A. Hagiescu, D. Singh, Gzip on a Chip: High Performance Lossless Data Compression on FPGAs using OpenCL, IWOC'14,

- Implementations of DELFATE:
(LZ77 algorithm and Huffman coding)
- Used in GZIP

Comparison:

- IBM Verilog on Altera StratixV A7 FPGA
- Intel Core™ i5 650 processor at 3.2 GHz
- OpenCL on a 28-nm Stratix-V A7 FPGA
- FPGA running at 160Mhz
- Processing 1 byte requires 9 cycles on i5
- FPGA processes 9 bytes per cycle

	Performance	Performance per Watt	Compression Ratio
OpenCL FPGA	2.84 GB/s	116 MB/J	2.17×
Intel Gzip	338 MB/s	9.26 MB/J	2.18×
Gap	8.5× faster	12× better	on par

	Performance	Efficiency	Productivity
OpenCL Kernel	2.84 GB/s 193 MHz	47% logic 70% RAM	High (1 month)
Verilog	3.0 GB/s 200 MHz	45% logic* 45% RAM*	Low
Gap	5.3% slower	2% more 25% more	--



3) Reducing power&size of operators

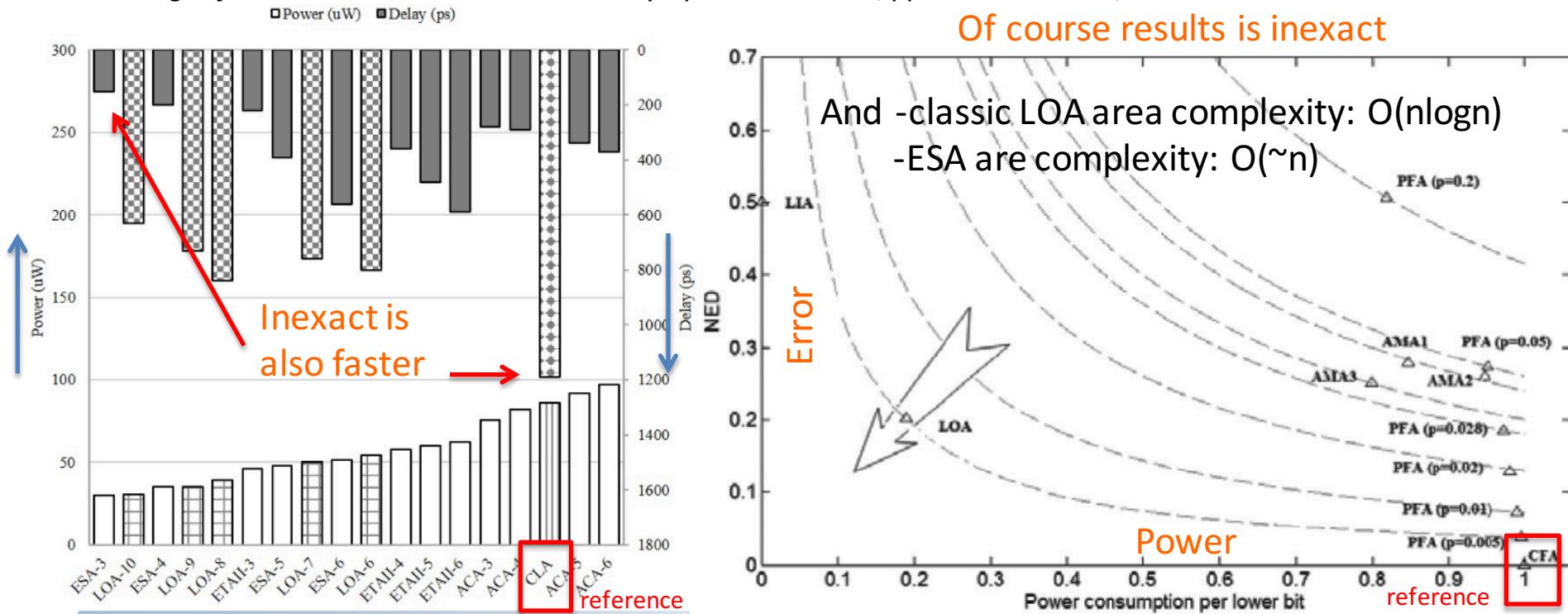
If Frequency, Power, and Integration are bounded

We can still customize the operators for the data format

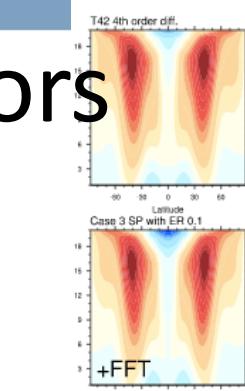
Approximate/inexact computing opens potential of significant power and size reduction. Example with integer adders:

Equal Segmentation Adder (ESA), Lower-part-OR Adder (LOA), Error-Tolerant Adder type II (ETAI), Almost Correct Adder (ACA), CLA: Carry Look Ahead Adder (full adder)

Jiang, Honglan, Jie Han, and Fabrizio Lombardi. "A Comparative Review and Evaluation of Approximate Adders." In *Proceedings of the 25th edition on Great Lakes Symposium on VLSI*, pp. 343-348. ACM, 2015.



3) Reducing power&size of operators



Can we really compute scientific applications with approximate/inexact floating point hardware?

Example1: Krishna Palem's probabilistic pruning and Climate simulation

- Compared the solutions of reference and pruned hardware for Lorenz'96 toy model (emulate the behavior of a meteorological variable X_n in a circle of latitude)

H1 versus Control:

	mean of X_n	Hellinger distance
A4: 1/2 area, power	control	3.77
M4: 1/3 area, power	H1	$3.78 \quad 1.03 \times 10^{-3}$

Düben, Peter D., Joven, Jaume, Lingamneni, Avinash, et al., . "On the use of inexact, pruned hardware in atmospheric modelling." Philosophical Transactions of the Royal Society A: Mathematical, Physical & Engineering Sciences, 372, no. 2018

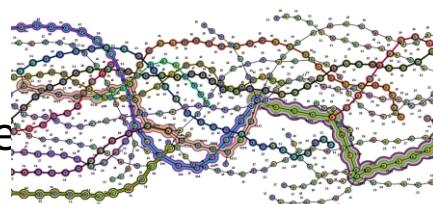
Exact computing on specific data sizes (not approximate)

(Bio-informatics: Genome Assembly & Alignment-free Sequence Clustering)

The four nucleotides can be represented in two bits,

And all amino acids can be represented in five bits.

Random Memory access for many application (caches are useless)



4) FPGA are already adopted

Microsoft adds FPGA in BING server to accelerate Bing Ranking

Microsoft Catapult

In ISCA 2014

We decided not to incorporate GPUs because the current power requirements of high-end GPUs are too high for conventional datacenter servers...

The added FPGA compute boards only increased power consumption by 10% and did not exceed our 30% limit in the total cost of ownership of an individual server.

Gain: ~X 2 throughput compared to software

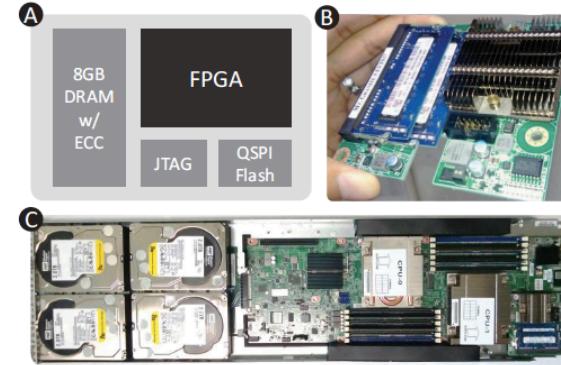


Figure 1: (a) A block diagram of the FPGA board. (b) A picture of the manufactured board. (c) A diagram of the 1 U, half-width server that hosts the FPGA board. The air flows from the left to the right, leaving the FPGA in the exhaust of both CPUs.

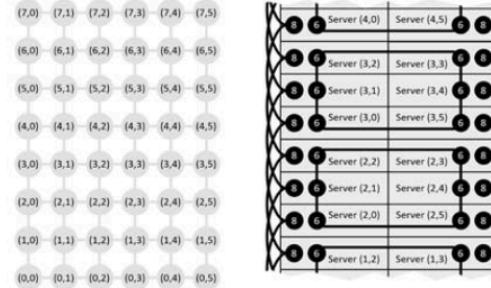


Figure 2: The logical mapping of the torus network, and the physical wiring on a pod of 2 x 24 servers.

Microsoft is also testing FPGA for multi-layer convolutional neural networks for non-trivial recognition tasks: large-category image classification and automatic speech recognition



4) FPGA are already adopted

Why is this important for the viability of FPGA in scientific simulation and data analytics (HPC and big Data)?

To Compete with CPU (PC, smart phone, tablet market) and GPU (Gaming market)

High-end FPGAs need to strong support marker.

- Data centers seems to be a good one
- What else?

→ The gigantic autonomous vehicle market

The screenshot shows a job listing on the Ford website. At the top, there is a navigation bar with the Ford logo and the slogan "Go Further". Below the navigation bar, the URL "Careers Home > Back to Previous > Ford FPGA Design Engineer" is visible. The main title of the job is "FPGA Design Engineer - Autonomous Vehicle - Virtual Driver Systems". The job ID is "Auto Req ID: 23736BR". The job title is "FPGA Design Engineer - Autonomous Vehicle - Virtual Driver Systems". The location is "Dearborn". The state is "MI". The company is "Ford Motor Company". There is a blue "Apply" button. In the top right corner of the page, there are social media sharing icons for LinkedIn, Twitter, Facebook, and Email, along with a small number "2". At the bottom of the page, there is a "Job Description" section which includes a detailed description of the responsibilities of the FPGA Design Engineer.

Applicability: Success Stories

Outstanding results:

- Grape-9 computer (U. Tokyo): 4 times higher Gflops/watt value than GPUs of the same generation for gravity computation.
- 3D-FFT on a cluster of directly connected FPGAs (Altera Stratix V): similar performance as the original Anton machine. Good results on molecular dynamics simulations and sequence analysis (Martin Herbordt at Boston University).
- BLAS2 kernel with comparable performance with 3-300 times less energy / GPUs and CPU.
- Sparse matrix computation on FPGAs with 3.5-6.5 speedups/optimized CPU implementation.



Other good results in:

Ok But all of them required HDL level programing.

Not really applicable to HPC and large applications
(HACC, Nek5000, FLASH, CESM, etc.)



Success Stories with OpenCL (2 examples)

Edge detection (image processing)

Center of High-Performance Reconfigurable Computing,
University of Florida

Results

OpenCL vs. VHDL productivity table		
	VHDL development time	OpenCL development time
Sobel, Canny, & SURF	6 months	1 month

Apps.	VHDL performance				OpenCL performance	
	Stratix 4		Predicted Stratix 5		Stratix 5	
	Frames/sec	Max freq.	Frames/sec	Max freq.	Frames/sec	Max freq.
Sobel	475	170	909	300	870	300
Canny	470	170	890	300	823	309
SURF	392	170	870	300	804	283

6 times faster development for 10% performance loss vs. VHDL

AES encryption

AES algorithm consists of multiple bit shifts and XOR operations
→ ideal candidate for FPGAs

Nallatech white paper

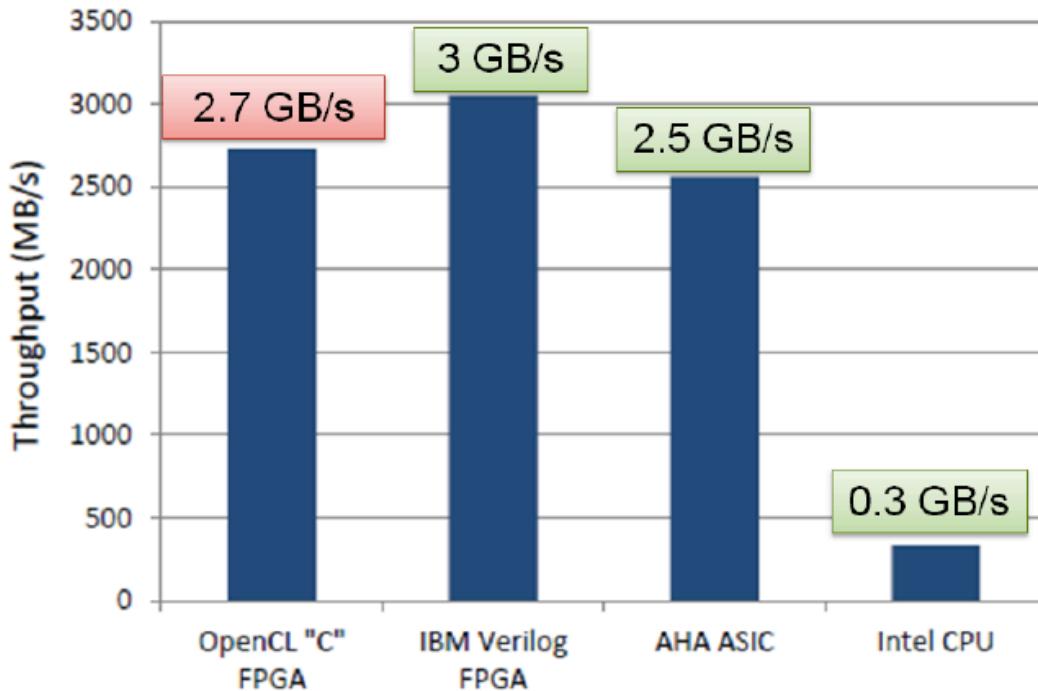
Technology	Throughput (GBytes/Sec)
E5503 Xeon Processor	0.01 (Single core)
AMD Radeon HD 7970	0.33
PCIe385 FPGA Accelerator	5.20

Several orders of magnitude faster than CPUs or GPUs

From Altera

Interesting example: Gzip with OpenCL

Case Study: GZIP Compression



Versus Verilog

OpenCL Was

10% Slower

12% more resources

3x faster development time

- Altera summer intern ported and optimized GZIP algorithm in a little more than a month
- Industry leading companies FPGA engineer coded Verilog in 3 months

Much lower design effort and design time



Reality Check (1/5)

If we go into the details:

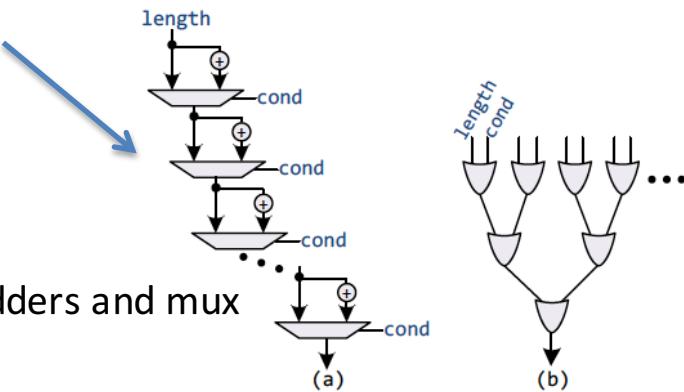
Authors used multiple tricks (at the C code level) to optimize the hardware produced by the OpenCL compiler (and the rest of the tool chain)

Example:

```
//compare current/comparison windows
#pragma unroll
for(char k = 0; k < LEN; k++) {
    if(curr_wind[j+k]==comp_wind[k][i][j] && !done[j])
        length[j]++;
    else
        done[j] = 1;
}
```

Because:

Long string of adders and mux



```
//compare current/comparison windows
#pragma unroll
for(char k = 0; k < LEN; k++) {
    if(curr_window[j+k]==comp_window[k][i][j])
        length_bool[i][j] |= 1 << k;
}
```

This writing allows the compiler to create a balanced tree of OR gates that uses lower FPGA resources.
(-25% of the total Gzip area)

BTW this is not how one writes such loop for a CPU (will rather use a while loop)

- 1) One needs to be expert in hardware, FPGA, and compiler to optimize the code
- 2) Verilog (IBM implementation) still 2X faster OR 2X smaller (area)

Reality Check (2/5)

The Re-Form project at ANL:

Focus on Scientific applications and data analytics

Objectives of the project:

- 1) Provide high level synthesis from OpenMP4 (accelerator directives)
- 2) Provide transparently hardware optimizations (no significant code modification)

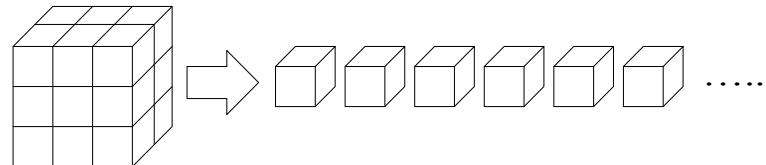
Altera state of the art Arria 10 FPGA on a dual socket Intel server:



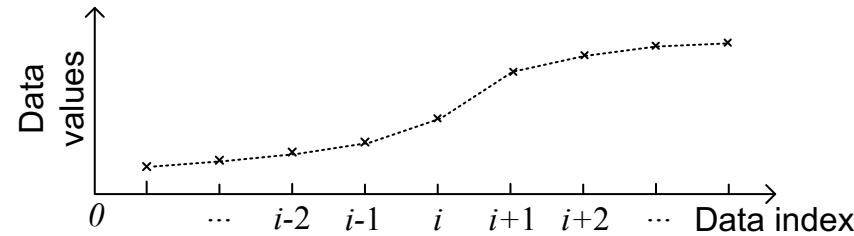
Reality Check (3/5)

First test: port the Argonne SZ lossy compressor for scientific data (best in class, as of today)

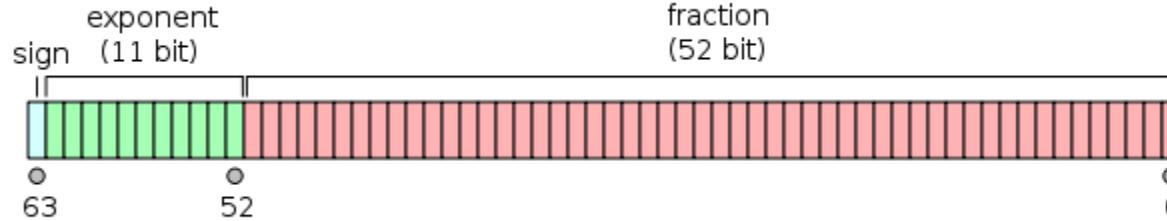
Step 1. Data linearization: Convert N-D data to 1-D sequence



Step 2. Approximate/Predict each data point by the best-fit curve-fitting models



Step 3. Compress unpredictable data by binary analysis



Step 4. Perform lossless compression (Gzip): LZ77, Huffman coding

Steps 1-3 prepare for strong Gzip compression

Reality Check (4/5)

- Simple implementation of SZ with OpenCL (making it a kernel and calling it from the host). **No optimization, no parallelism.**
- Initial code ~1000 lines of C.
- Use the Altera OpenCL compiler:
 - Remove dynamic memory allocation
 - OpenCL 1.2 does not allow double pointers
 - The host code and the FPGA code works on the emulator (Yes!)
 - But emulator only takes OpenCL code and verifies that the OpenCL code works functionally: nothing to do with FPGA
 - 80 minutes to compile from OpenCL to Verilog



Reality Check (5/5)

→ Compilation crashes before completion (after several hours!!!)

- Culprit: the 3rd step that performs binary analysis.
- Root cause: Large local arrays are allocated and there is not enough BRAMs (Bloc RAMs) to handle them.
- Remove local function variables (at least the large ones: 1MB) or replace by global value. OR make it fit in 1 block RAM (1KB)
- 3MB of BRAM in the Arria 10 FPGA (max on Xilinx: 9MB)
- Reduce the code to the second step (prediction): 500 lines of code.
- The total compilation time is 5 hours on Duterros [CPU@2.60GHz](#) (16 cores)
- FPGA resource usage: 5% LTUs, 7% registers; 12% memory blocks; 2% DSP blocks. In total, the logic utilization is 12%.
- **BUT Board interface consumes a lot of resource.**
 - SZ step 2 consumes only 3% of the resources!
- Frequency of the FPGA: ~318 Mhz.
- Started this 2 weeks ago with several experts in FPGA.
- **Conclusion: It will take time even for a simple compressor (1K lines).**



Conclusion

- New FPGA SoC feature very attractive set of features: DSP, advanced FPGA core fabric, multi-core CPUs, large number of Transceivers
- Several additional trends make FPGA attractive for HPC and data analytics: compression, power and size
- Massive adoption is key to success: Data centers & autonomous vehicles
- Several success stories showing clear advantages of FPGAs over GPUs and CPUs (performance, power)
- However, Not ready for prime time in HPC (needs more research)

Christopher Fenton
Put a CRAY1 in FPGA in
2010, now @ D.E. Shaw



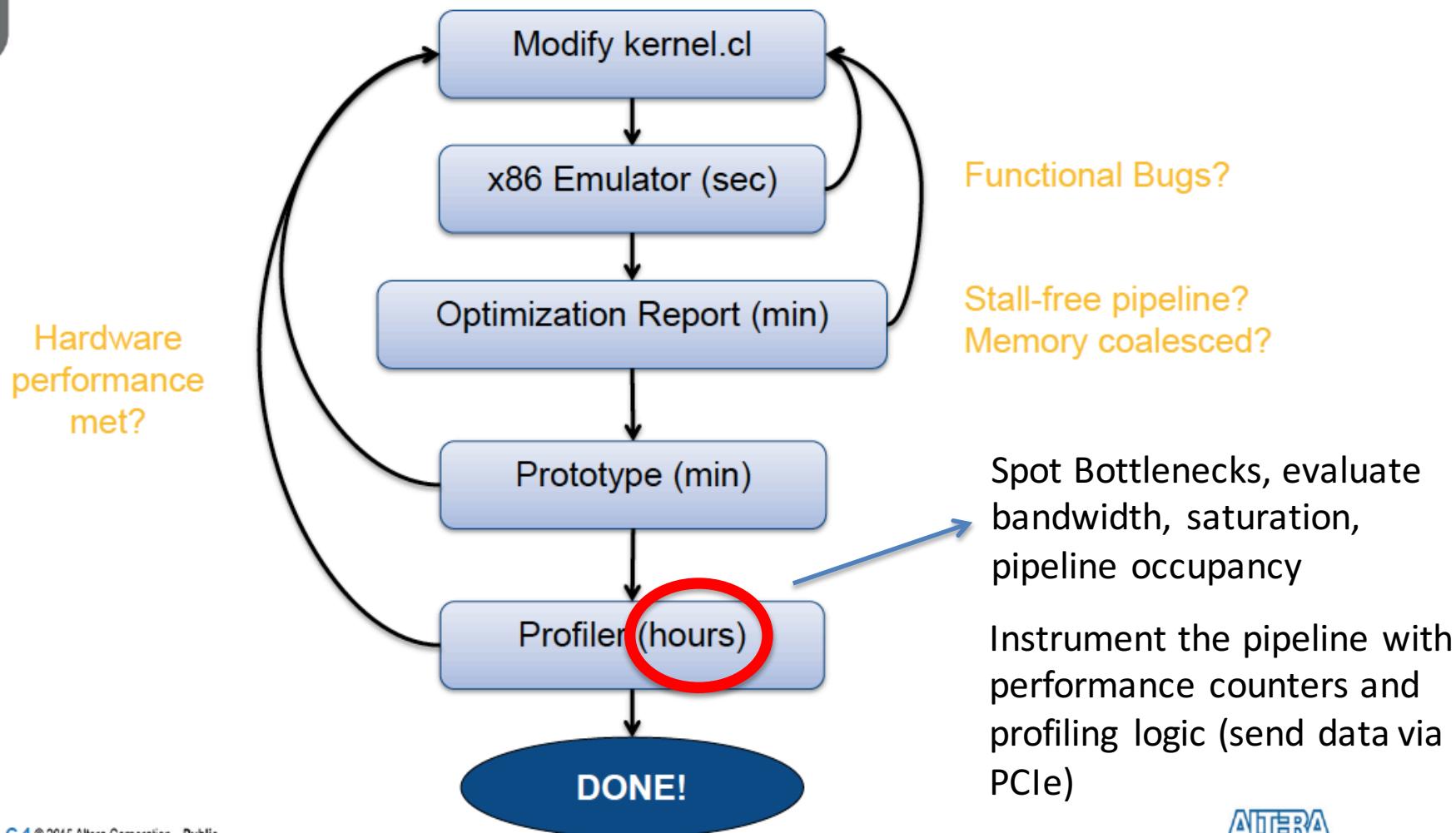
Classic Hardware Configuration Chain

- OpenACC, OpenMP
- OpenCL, C, C++
- High-Level Synthesis (or Behavioral synthesis before)
- Language-based design: Register-Transfer Level (RTL). VHDL / Verilog
- MSI-level design (adders, multipliers, registers)
- Gate-level design (ANDs, ORs, and NANDs)



Compilation/optimization time is a problem

Kernel Development Flow

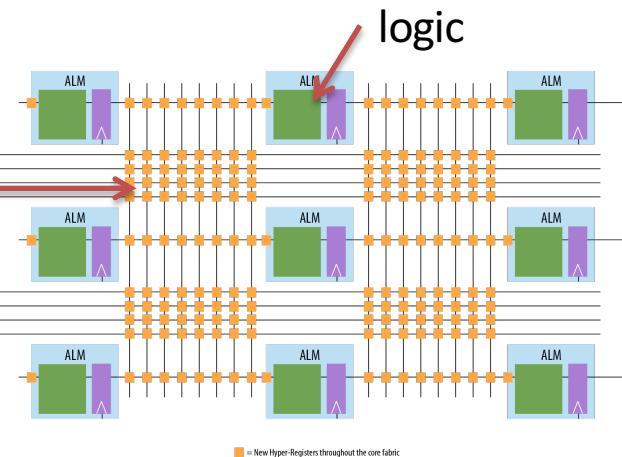


1) New FPGA are getting competitive (against GPUs and CPUs)

What really could be game changer:

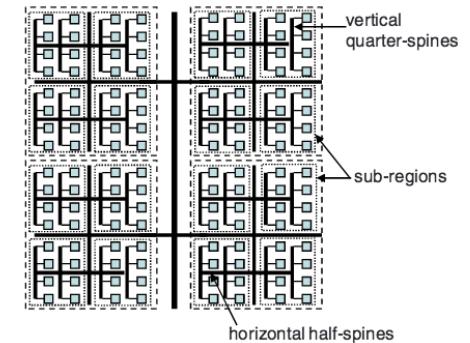
- In core fabric bypassable registers

A Register associated with each routing segment
A way to reduce clock period in the FPGA fabric
(clock depends on the longest section between
flipflops. Having in core fabric register allows
reducing the longest section and balance sections lengths)



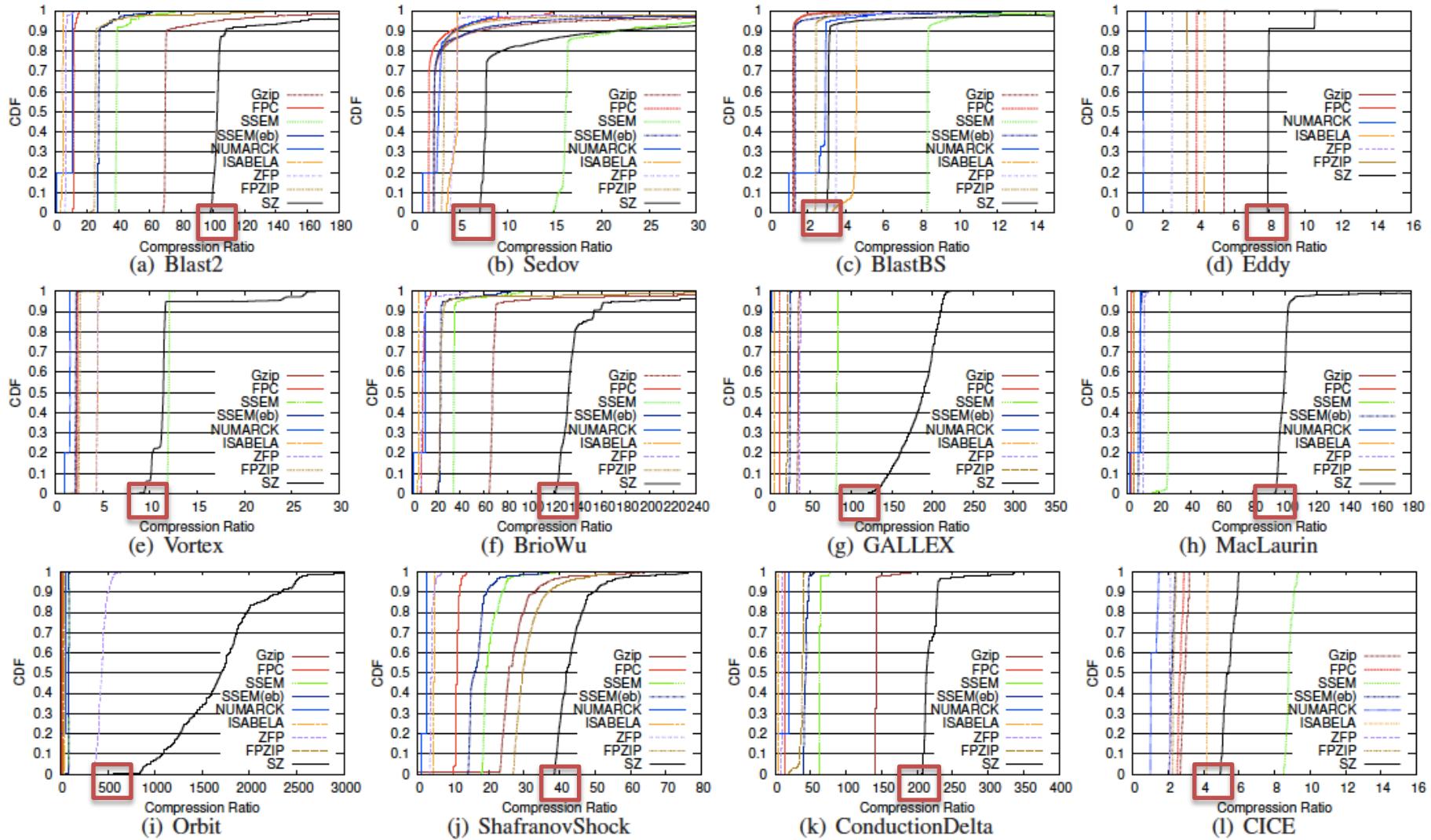
- Programmable clock trees

Clocks only synthesized where needed,
to minimize dynamic power
(Clocks are the single largest source of dynamic power usage)



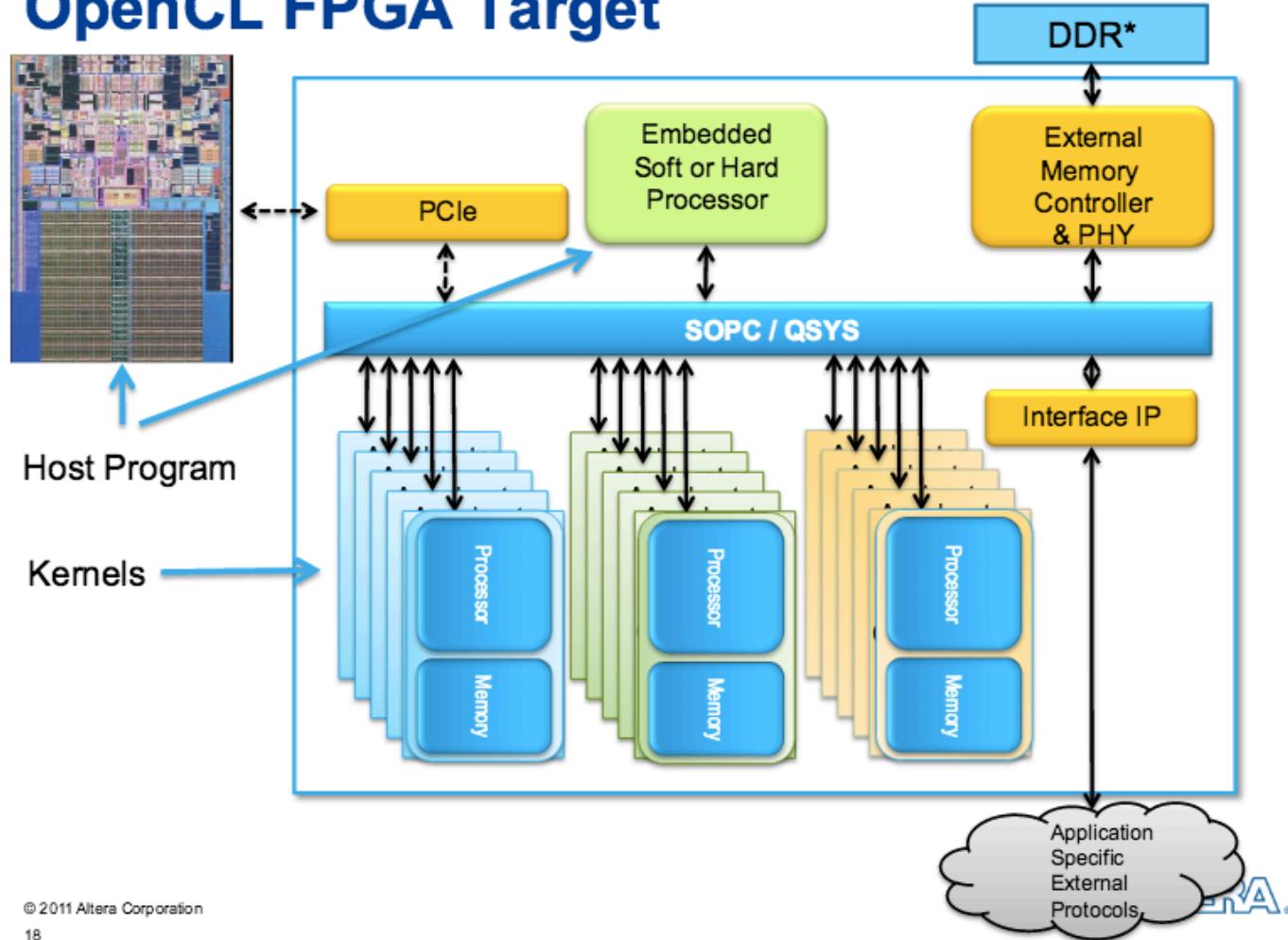
2) Data as the main perf. constraint

Lossy compression is much more promising for scientific computing (not for checkpointing but for post analysis)

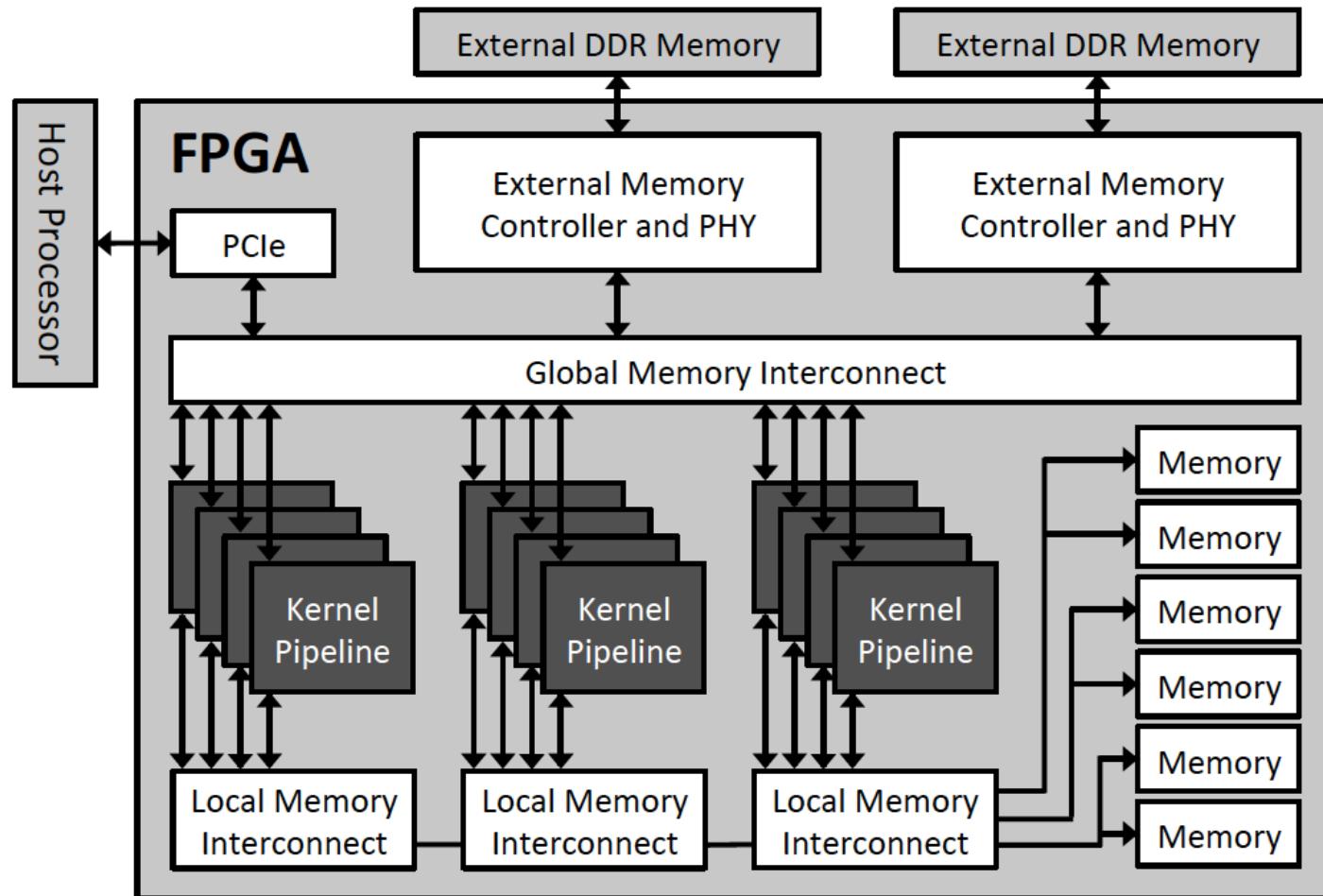


Running OpenCL codes on FPGA

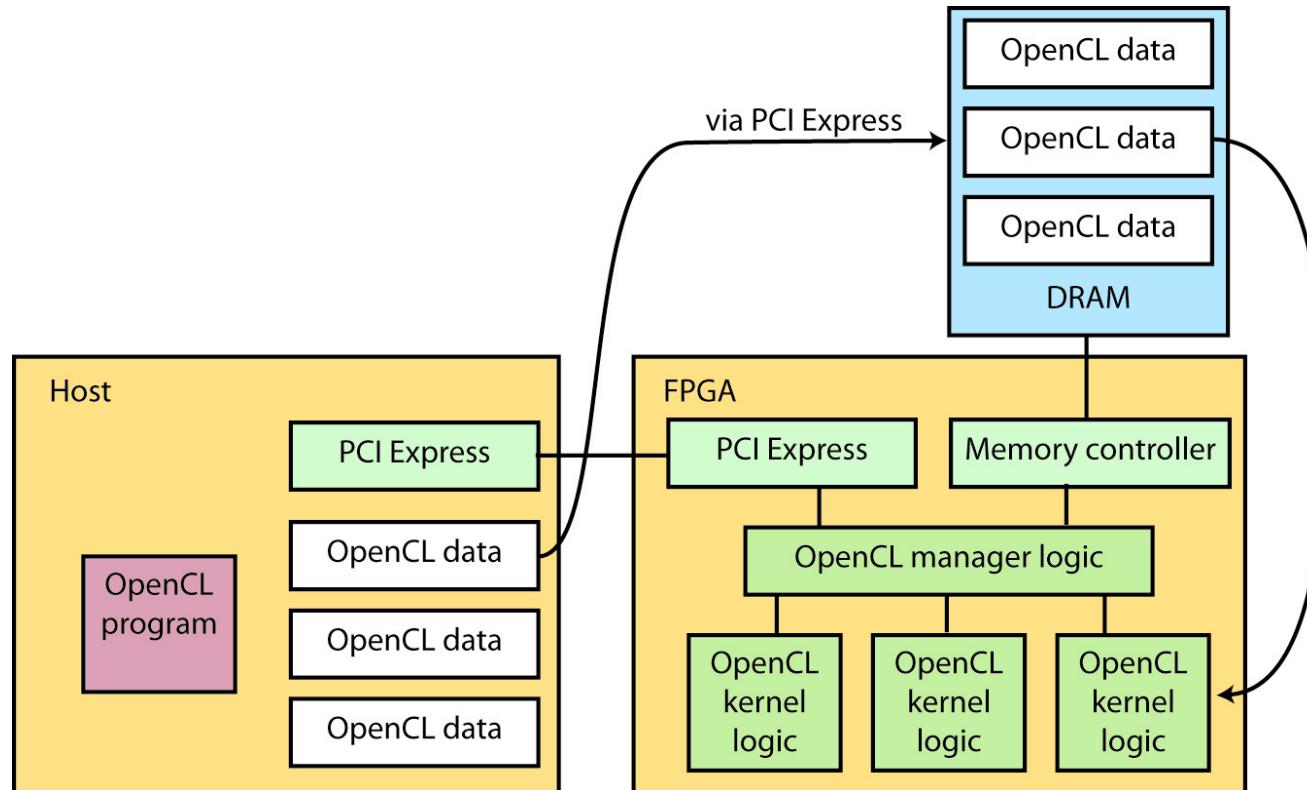
OpenCL FPGA Target



Running OpenCL codes on FPGA



Running OpenCL codes on FPGA



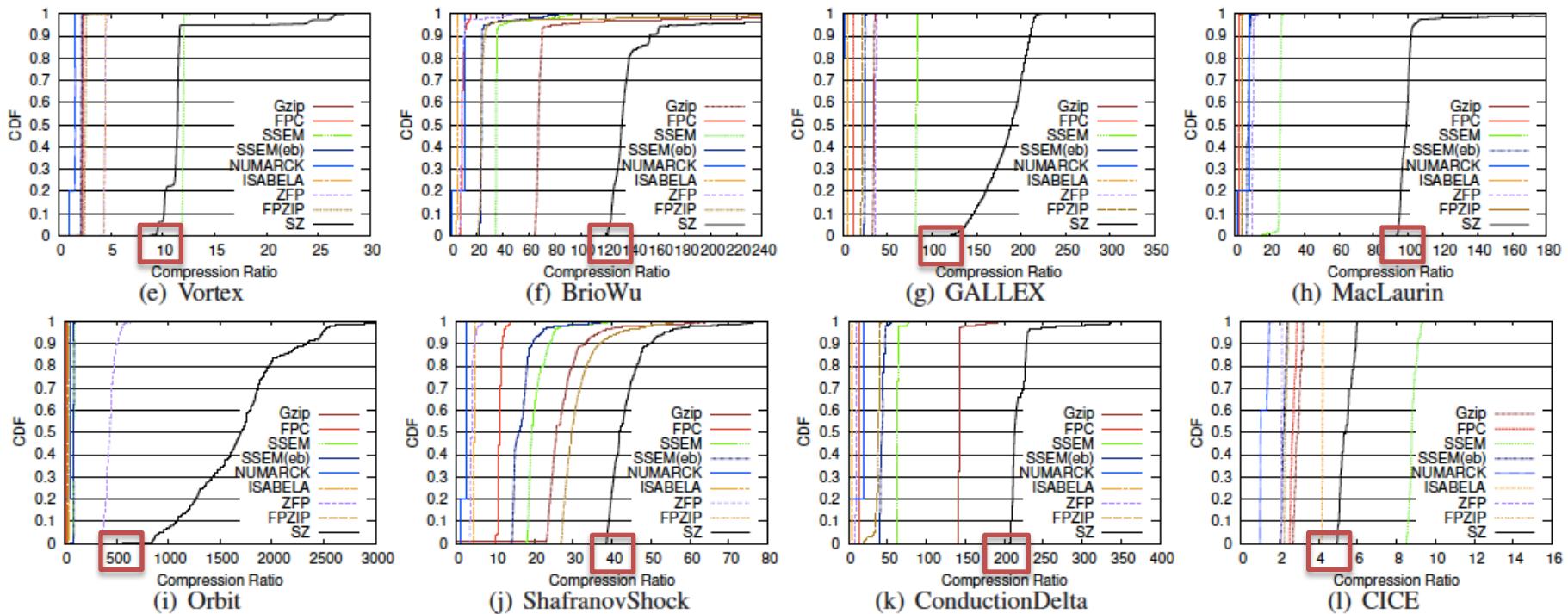
2) Data as the main perf. constraint

Lossy compression (example)

ANL SZ: “Simple” lossy compressor leveraging data smoothness:

- Linearize data
- Try 3 fitting functions (linear, quadratic interpolation or identical)
 - If not store the full data
 - If success, store 2 bits describing the type of interpolation
- compress the 2 bit sting by Gzip

Simple operations,
Highly parallel,
well suited
for FPGA



2) Data as the main constraint

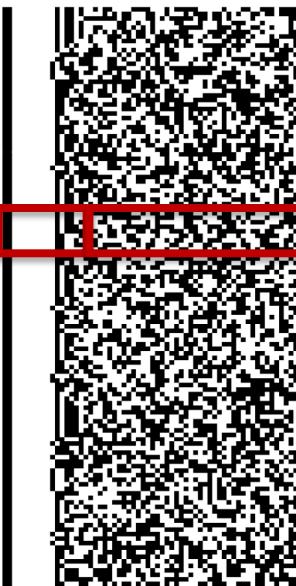
Compression is needed.

- Floating point data sets

This is what we need to compress
(bit map of 128 floating point numbers)

Sign+
Exponent

Floating point data set
(numerical simulation
of the brain):



Mantissa

Random
(noise)



Image from Leonardo Bautista Gomez (BSC)

- Floating point data sets looks very random at first sight (except for the sign and the exponent)