FPGA-based Supercomputing: New Opportunities and Challenges

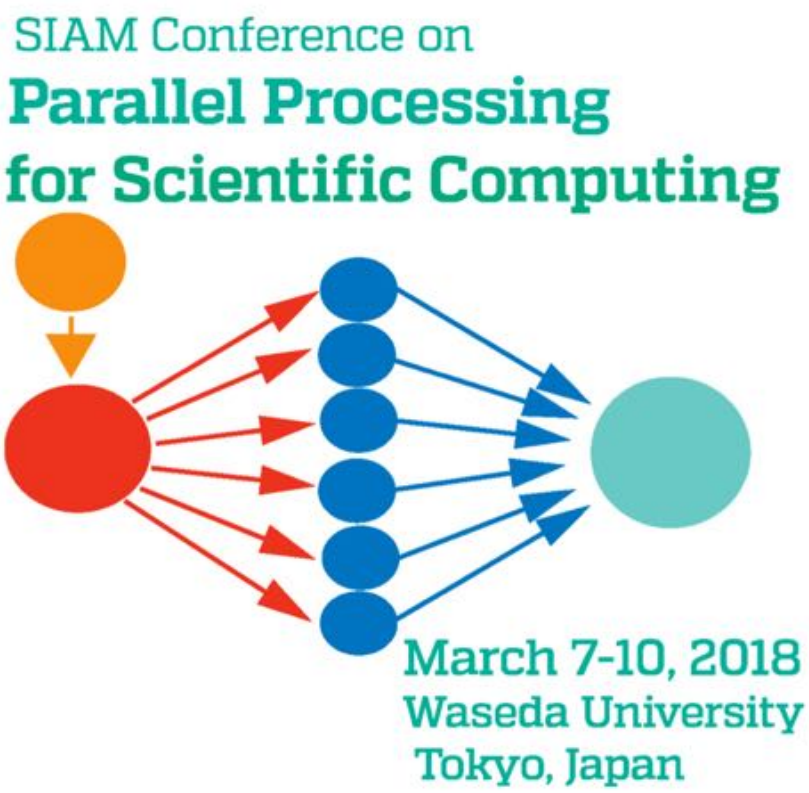
Naoya Maruyama (RIKEN AICS)*

5th ADAC Workshop

Feb 15, 2018

* Current Main affiliation is Lawrence Livermore National Laboratory

SIAM PP18: Deep Learning + HPC



- MS 26: Deep Learning from HPC Perspectives
- Thursday, March 8
- Speakers from ORNL, ETH, University of Tokyo as well as LLNL, ANL, etc.

Why FPGA?

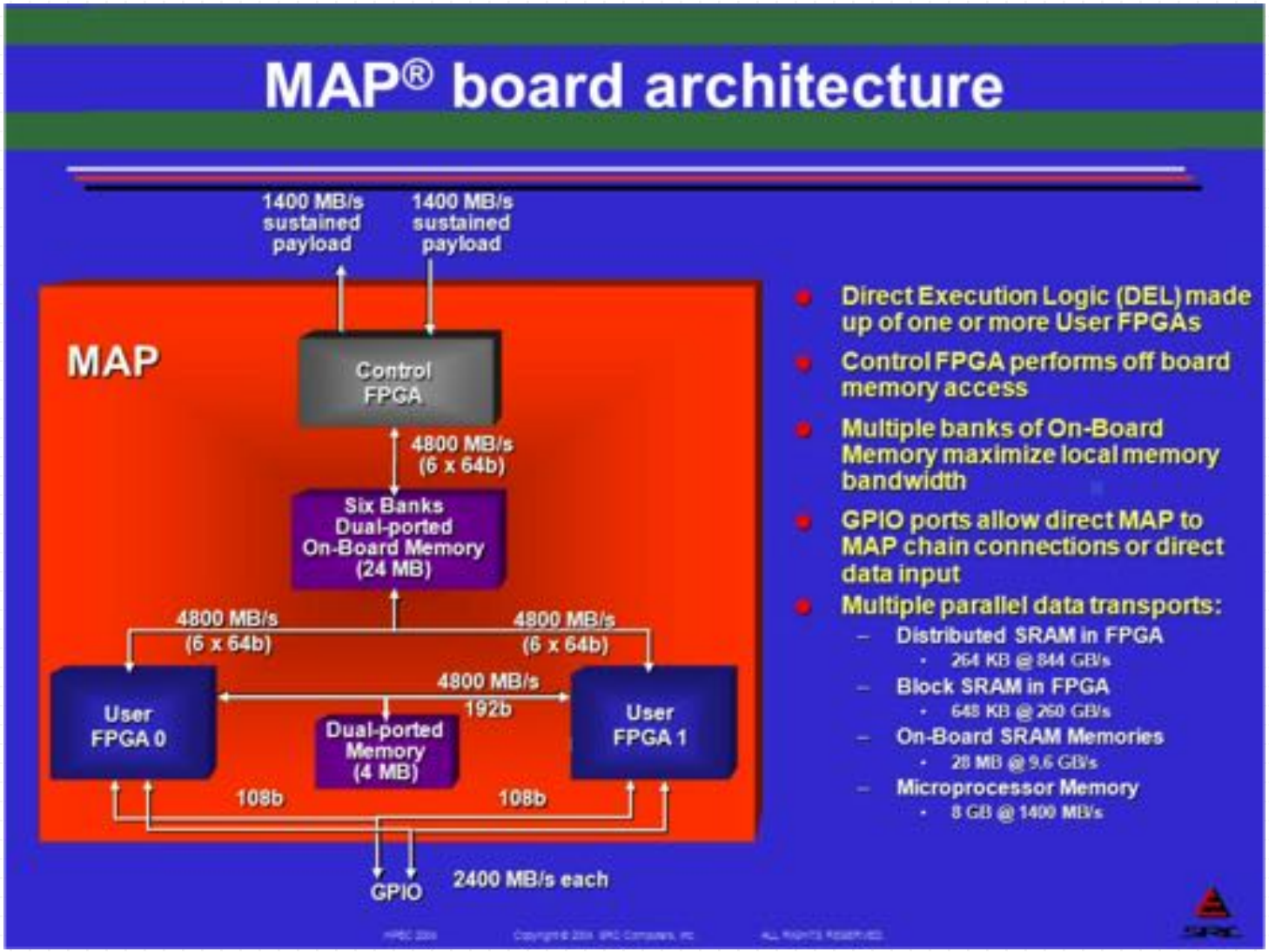
- Dark silicon is bringing specialized logic
  - TPU, Nervana, Graphcore, Tensor Core, …
  - Embedded AI/vision processing units in modern smartphones
- *But*, only to highly profitable and uniform application domains
- HPC is:
  - Not immediately profitable
  - Very diverse
- FPGA is a middle-ground solution
  - Not quite efficient as ASICs, but reconfigurability makes specialization possible for such application domains as HPC
  - Growing non-HPC markets that HPC could leverage
    - FPGA by default at Microsoft datacenters
  - *Poor man's specialization*

But Why Now Again?

- Significant efforts till late 00's
  - SRC Map system, Cray XD
- Largely dwarfed by the emergence of GPGPU

- Why now?
  - Competing hardware performance
    - Large arrays of IEEE FP-capable DSPs (e.g., 10 TFLOPS in SP of Stratix 10)
    - HBM2-equipped boards by both Altera and Xilinx
  - Lower psychological and practical barrier for accelerator-based computing thanks to the success of GPU
  - More mature and standardized programming environments
    - OpenCL compilers for FPGAs are provided by both Altera and Xilinx

SRC-6 MAP System (Circa 2004)



Source: SRC slides at HPEC 2004

facebook.com

facebook

Email or Phone

Password

Log In

Forgot account?

SRC Computers, LLC

@srccomputers

Home

About

Photos

Posts

Community

Create a Page

SRC®

SRC COMPUTERS, LLC

Like Share ···

Watch Video

Send Message

Photos

Computer Repair Service in Colorado Springs, Colorado

Permanently Closed

Community See All

32 people like this

34 people follow this

About See All

+1 719-282-0213

www.srccomputers.com

Computer Repair Service · Electronics · Computer Company

COVER FEATURE

Using FPGA Devices to Accelerate Biomolecular Simulations

Sadaf R. Alam, Pratul K. Agarwal, Melissa C. Smith, and Jeffrey S. Vetter
Oak Ridge National Laboratory

David Caliga
SRC Computers

A field-programmable gate array implementation of a molecular dynamics simulation method reduces the microprocessor time-to-solution by a factor of three while using only high-level languages. The application speedup on FPGA devices increases with the problem size. The authors use a performance model to analyze the potential of simulating large-scale biological systems faster than many cluster-based supercomputing platforms.

Despite the tremendous capability, flexibility, and power efficiency of field-programmable gate arrays (FPGAs), their use in scientific high-performance computing has been largely limited to numerical functions and kernels imple-

ACHIEVING APPLICATION SPEEDUP

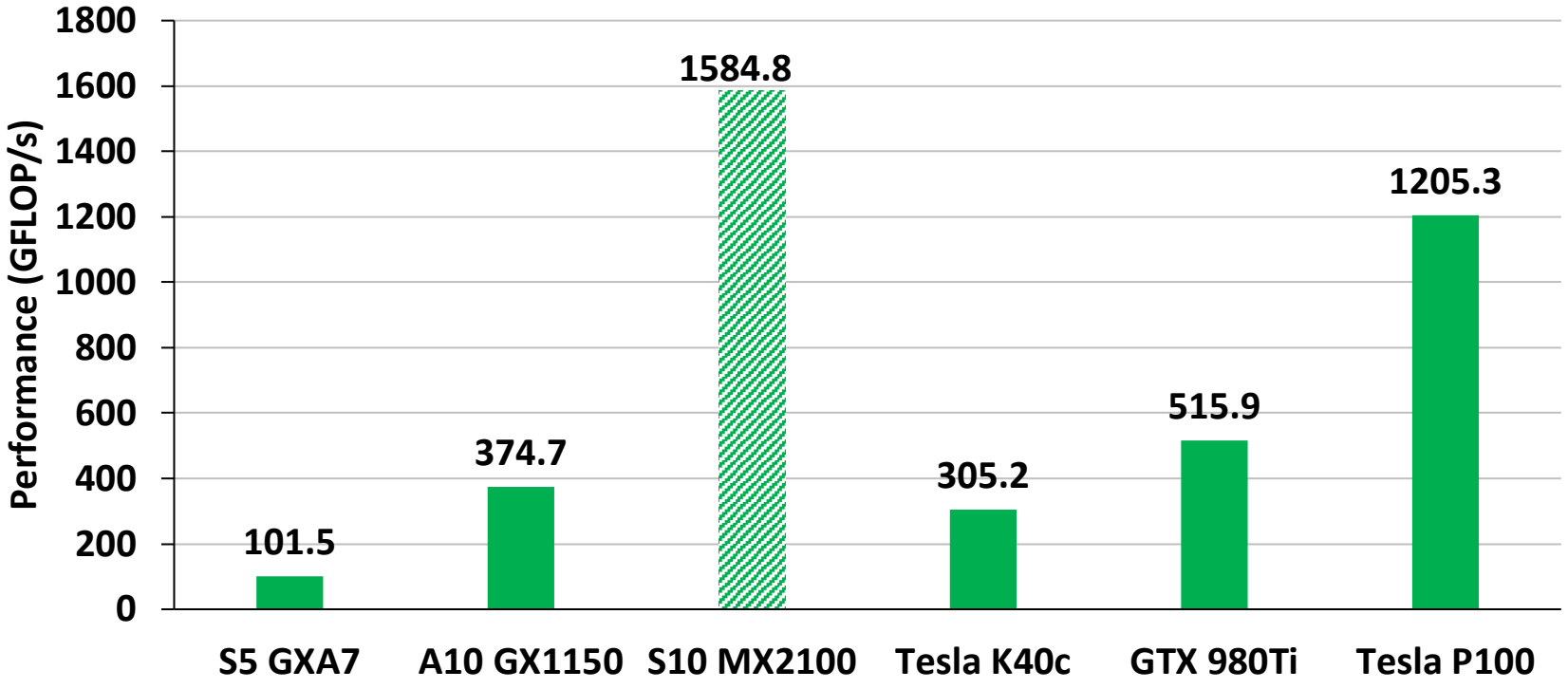
Researchers have employed various strategies to accelerate the PME calculations in their large-scale experimental simulations on traditional, parallel supercomputing platforms. Currently, even the fastest com-

But Why not GPU?

- If performance is bounded by memory bandwidth, compute reconfigurability won't matter for performance
- Many HPC codes are memory bound

- *Baseline*: Latest high-end FPGAs would be able to achieve comparable performance as that of GPUs for those apps
- *Opportunity*: Communication avoiding algorithms with FPGAs

Case Study: 3D Diffusion Stencil



- Zohouri, et al., " Combined Spatial and Temporal Blocking for High-Performance Stencil Computation on FPGAs Using OpenCL," FPGA'18, Feb 2018 (to appear)
- Aggressive temporal blocking applied for FPGAs: 4-way with S5 and 12-way with A10
- GPU also uses temporal blocking but only 2-way as the speedup diminished

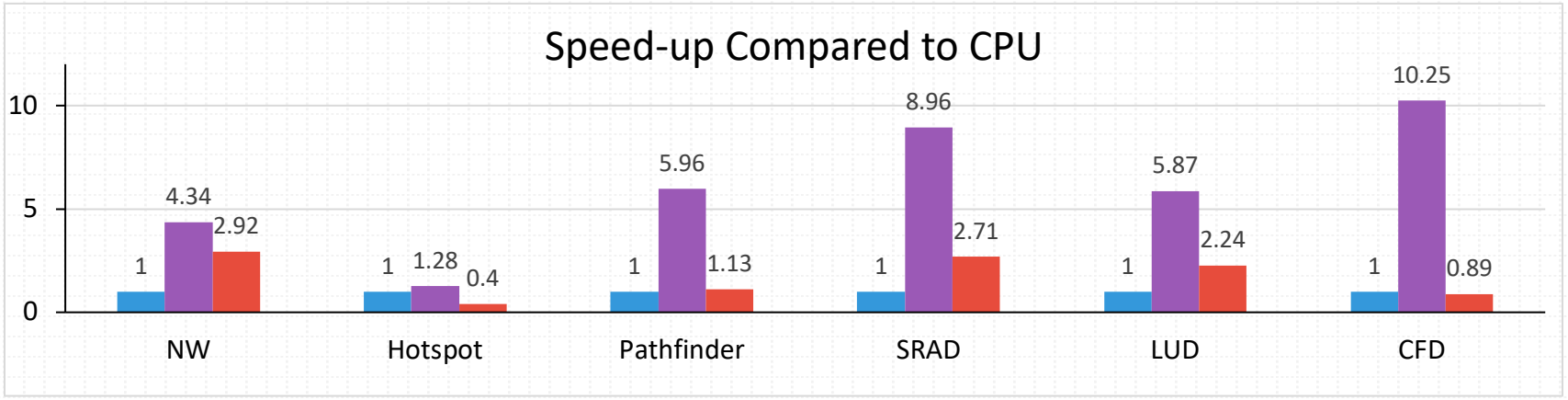
Research Agenda

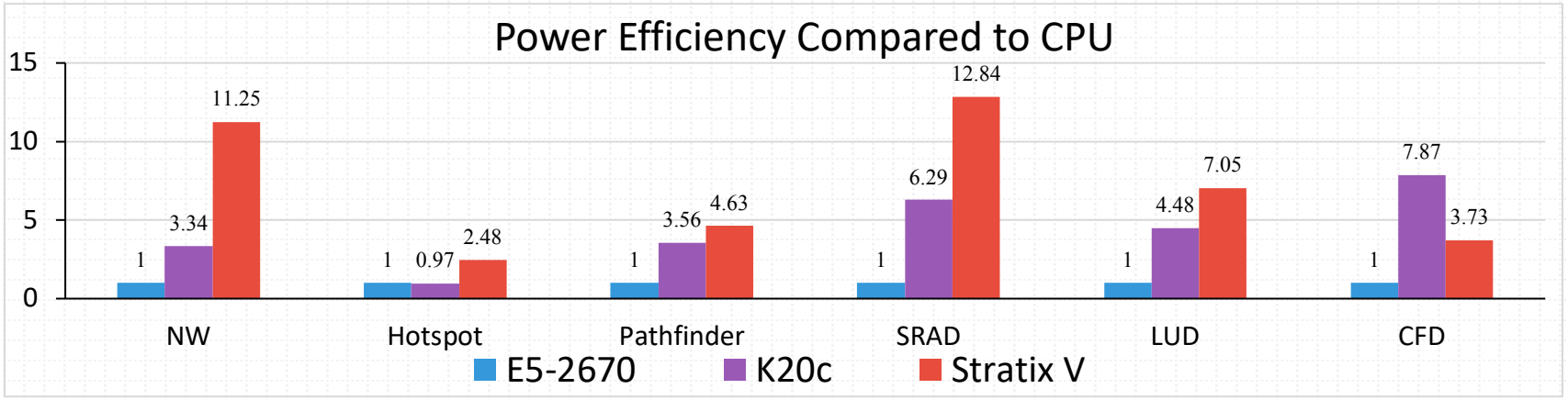
- Performance characterization
- Identifying performance optimization techniques
- Tools to support performance analysis and optimization
- Debugging and validation tools
- Compilation techniques for exploiting reconfigurable dataflow
- Extending existing accelerator-aware programming models for FPGAs
- Performance portability across FPGAs, GPUs, CPUs, …
- Multi-FPGA programming
- Fault tolerance
- Programming tools for variable precision
- …

Evaluating FPGAs with OpenCL

- Performance characterization of FPGA performance when used with OpenCL
- Used Altera (Intel) OpenCL compiler and the Rodinia benchmark suite [Che et al. 2009]
  - Consists of 23 benchmarks, each with OpenMP, CUDA and OpenCL versions
  - Covers the Berkeley Dwarfs
- Evaluation approach
  - Use the original OpenCL version as baseline
  - Convert the baseline to more FPGA-friendly programs
    - E.g., use shift register-based data forwarding rather than thread barriers
- Extended benchmark code is available at
  - https://github.com/fpga-opencl-benchmarks
- For more details, see:
  - H. Zohouri, et al., "Evaluating and Optimizing OpenCL Kernels for High Performance Computing with FPGAs," SC16.

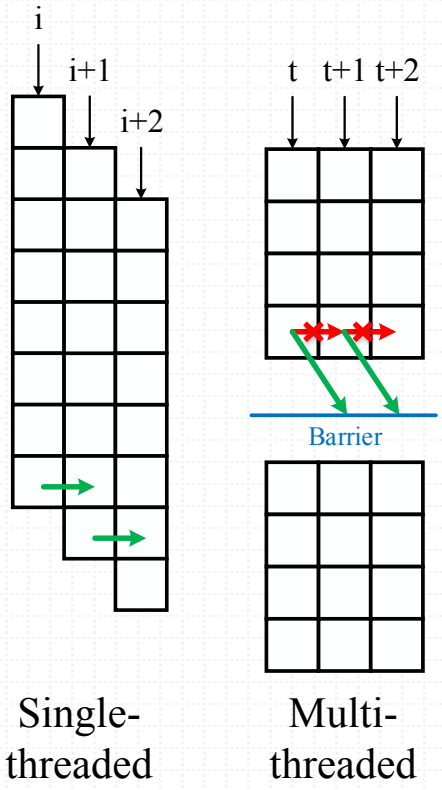
Performance and Power Comparison





Performance Portability Challenge

- On GPUs, OpenCL local memory ("shared memory" in CUDA) is used for data locality optimization
  - *Barriers* are necessary to guarantee local memory consistency
- Also possible on FPGAs with multi-threaded kernels but:
  - Barriers result in pipeline flush
  - No other way to pass data from one thread to another
- In single-threaded kernels:
  - Iterations are issued sequentially → Pipelined
  - Issue distance between iterations can be used to pass data
  - Single-clock register read/write allows passing data
- Dependencies are mostly penalty free, barriers aren't
  - Single-threaded is preferred over multi-threaded



Example: Rodinia SRAD

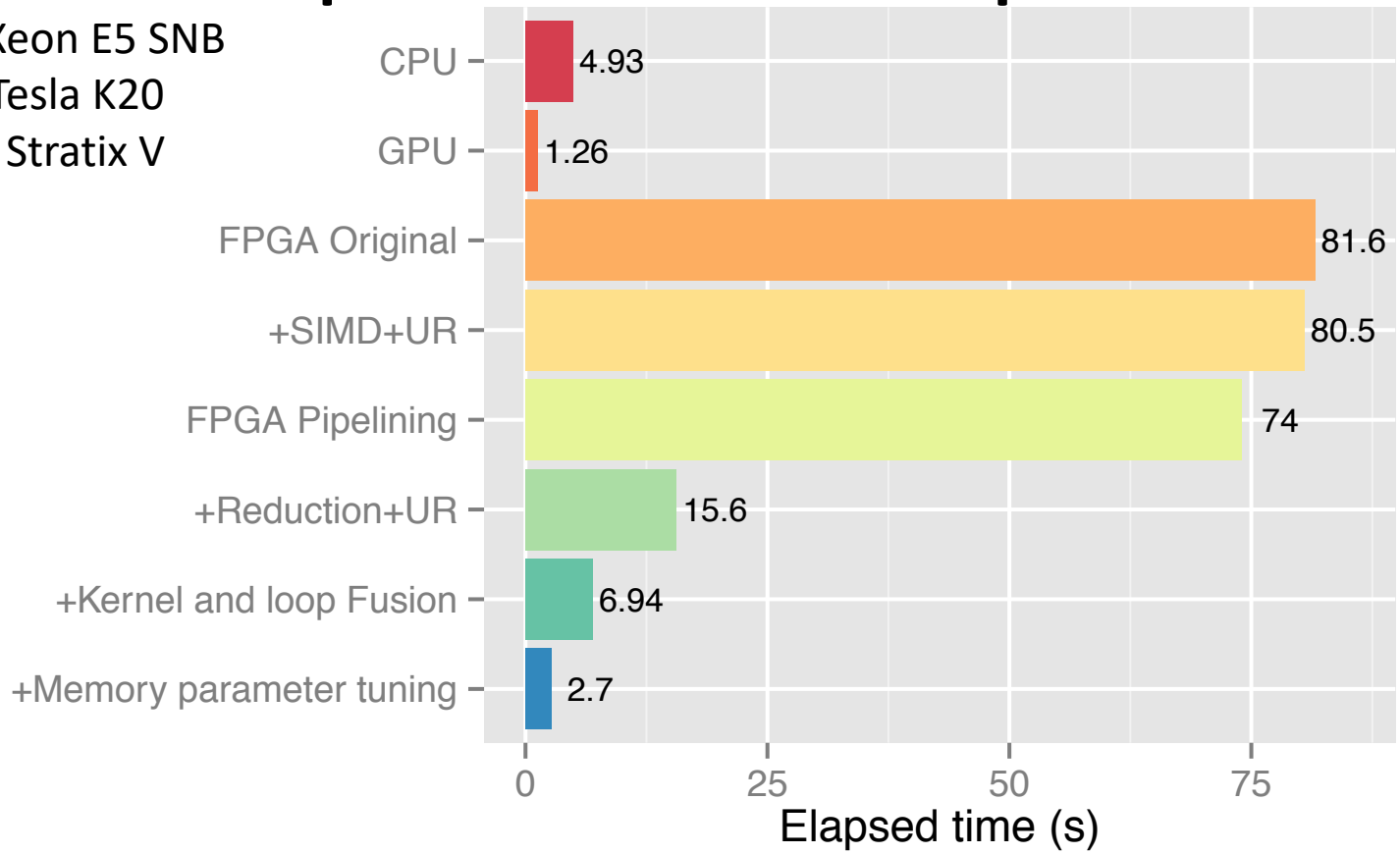
- 2 preparation kernel
- 4 compute kernels
  - 1: Copies (in) and $(in)^2$ to two buffers
  - 2: Performs reduction on the two buffers
  - 3: 5-point stencil computation (like Hotspot)
    - 5 input and 5 output off-chip buffers
  - 4: Dynamic programming computation (like NW)
    - 5 input and one output off-chip buffer
    - Iteration dependency

SRAD: Original Implementation (Cont.)

- Original implementation favors high memory bandwidth
  - Performs the computation in multiple passes over multiple kernels
  - Uses no local memory-based optimization except for reduction
  - Passes data between kernels using multiple global buffers
- Does not work well on FPGAs due to:
  - Low off-chip memory bandwidth
  - Multiple mathematically-complex kernels and limited area

SRAD: Optimization Impact

CPU: Xeon E5 SNB
GPU: Tesla K20
FPGA: Stratix V



- Kernel and loop fusion
  - Conversion to one kernel with 4 loops: 10% area reduction
  - Combine loops 1 & 2: Two less off-chip buffers, Less area
  - Allows memory optimization: Read neighbors from same buffer, 4 less off-chip buffers
  - Combine loops 3 & 4: Blocking + Sliding Window, 5 less off-chip buffers

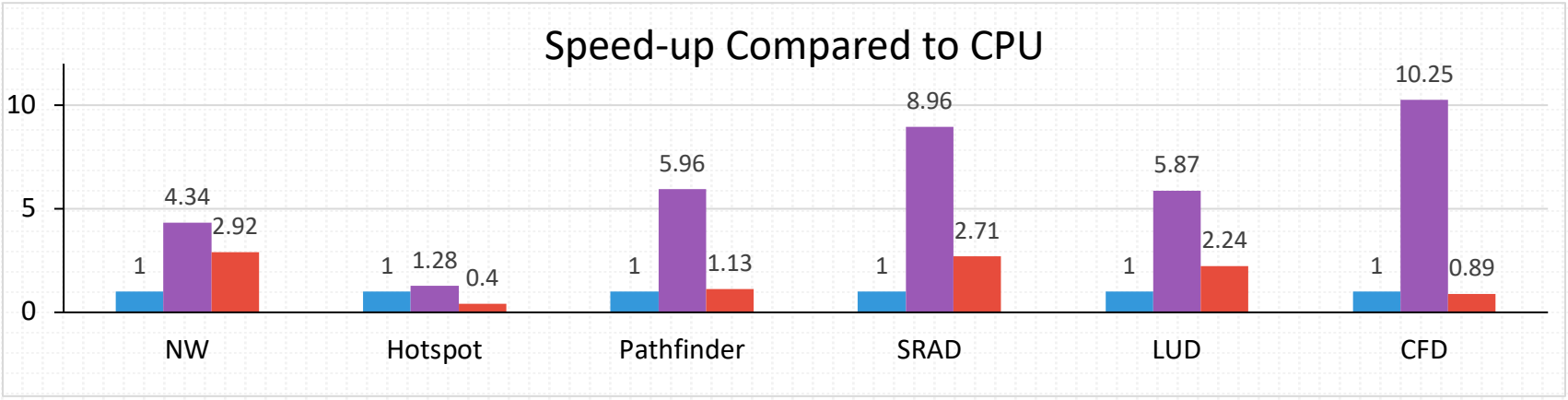
Example: Rodinia NW

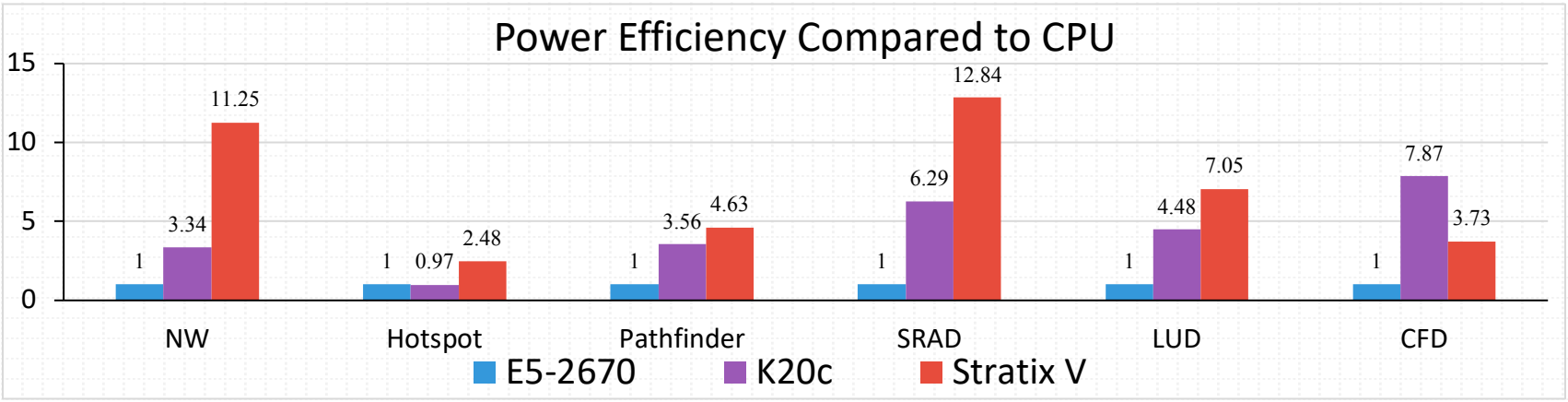
- Rodinia benchmark version 3.0
- Altera OpenCL v15 on Stratix V FPGA (Terasic DENET-5)

Type	Optimization	F_{max} (MHz)	Run Time (ms)	Power Dissipation (Watt)	Power Usage (J)
Thread	None	277.2	16574	12.01	199.1
Loop	None	243.4	117523	10.59	1245.2
Thread	Basic	194.7	2445	16.94	41.4
Loop	Basic	249.1	116457	9.93	1156.7
Loop	Advanced	148.0	251	15.44	3.8

Shift register optimization is 66x faster than baseline

Performance and Power Comparison





- More details in the paper
  - Analytical performance models
  - Optimization strategies

Next Steps

- Updating performance results with HBM2 memory
- Identifying when the FPGA can outperform the GPU
  - Promising results with simple stencils by aggressive CA techniques
- High-level programming systems
  - Automating FPGA-specific optimizations as compiler-based transformation
  - Extending overlay techniques for faster compilation