Evaluation of a Floating-point Intensive Kernel on FPGA
A Case Study of Geodesic Distance Kernel

Zheming Jin, Hal Finkel,
Kazutomo Yoshii (speaker), Franck Cappello

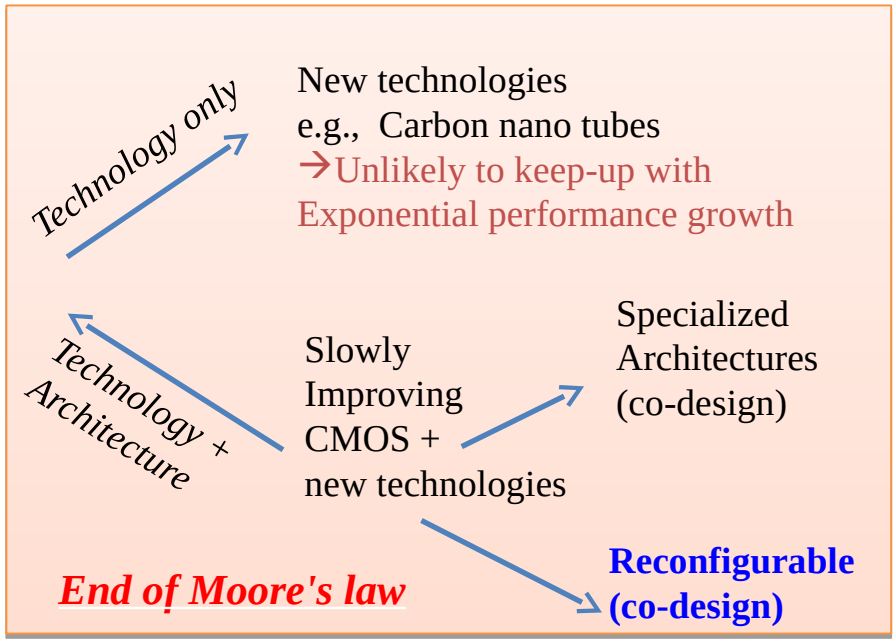
Argonne National Laboratory, Argonne, IL, USA

{zjin, hfinkel, kazutomo, cappello}@anl.gov

The 10th Workshop on UnConventional High Performance Computing (2017)
August 29, 2017, Santiago de Compostela, Spain

Re-form: Leveraging FPGA Reconfigurability and Floating-point Capabilities for Next-generation Computing Systems

- Exponential performance growth will end soon
  - New technologies may not be ready for deployment soon
- Specialized designs are expensive
  - e.g., H.264 encoder, Anton by D.E. Shaw, GRAPE architecture, etc
  - Impressive performance and energy efficiency but inflexible and higher nonrecurring engineering costs
- Leverage FPGAs for HPC
  - Funded by laboratory directed research and development (LDRD)
  - Address technical gaps
  - Prototype FPGA work flow for HPC



Programming FPGAs

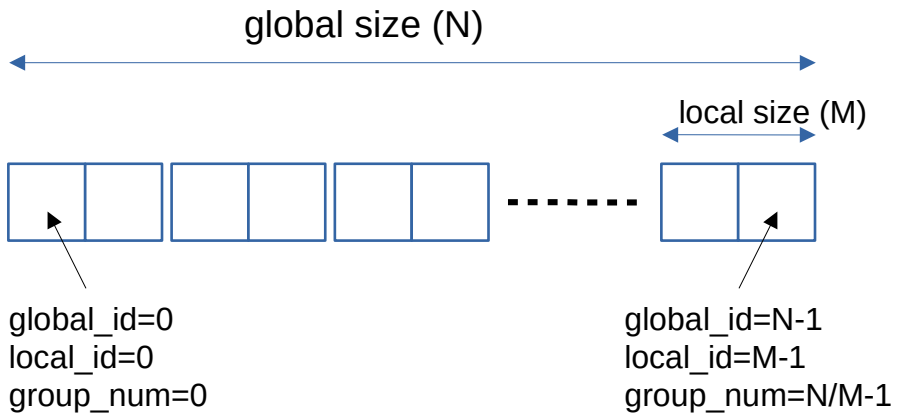
- Hardware Description Language (HDL)
  - Verilog, VHDL
  - Behavioral model, bit-level, dataflow nature, etc
  - Relatively longer and painful development, verification
  - Vendor tools translate HDL to configs (very slow!)

- High-Level Synthesis (HLS)
  - Accept higher-level language like C/C++
    - Vivado HLS, LegUp, OpenCL
    - Limitations
  - Custom Language
    - Maxeller (Java)
  - Convert to HDL

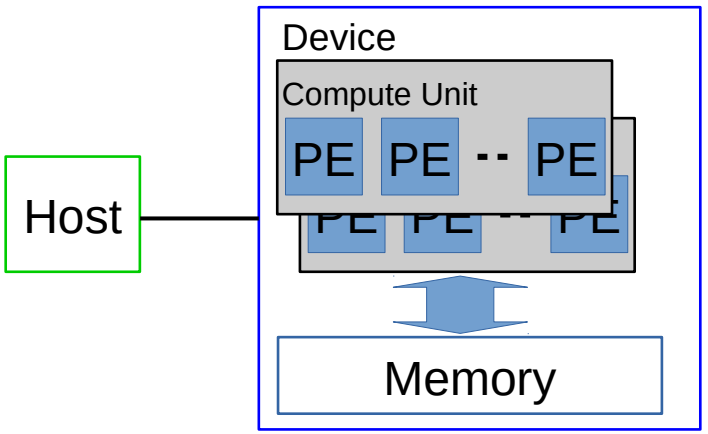
- Overlay, coarse-grain architecture

OpenCL for FPGAs

- Support data and task parallelism
  - Single-thread vs NDRange kernel
- Optimization space is larger than CPUs/GPUs
  - Due to reconfigurability
- Number of work-items
  - Conceptually a work-item is a thread
  - Pipeline parallelism on FPGAs
- Number of compute units
  - Single unit by default (Intel OpenCL)
- Loop unrolling
- SIMD

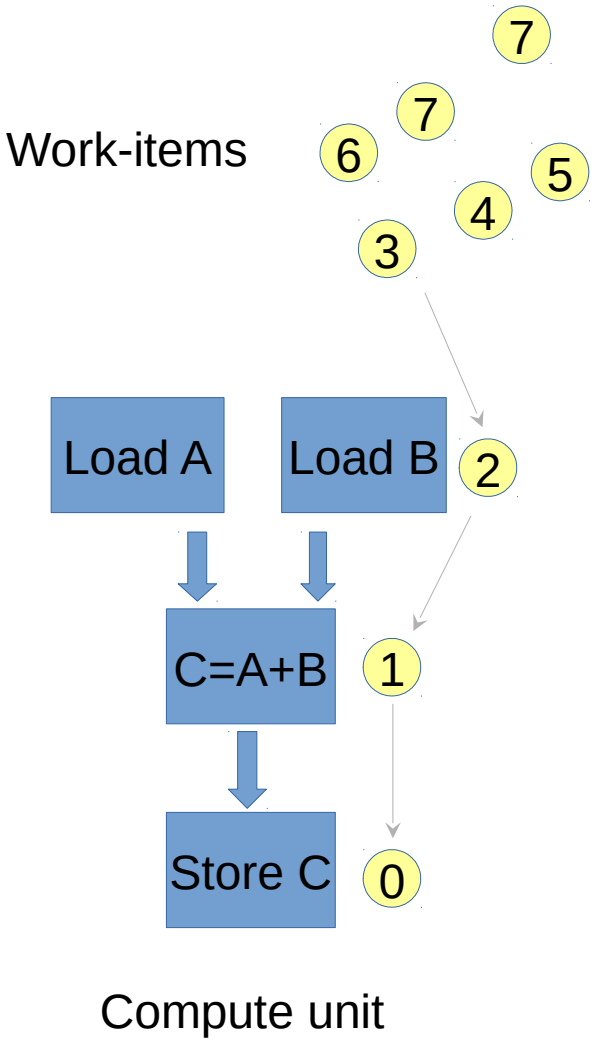


OpenCL data-parallel model: 1D example



OpenCL platform model: single device example

Pipeline Parallelism




```
__kernel void vector_add (
        global const double *x,
        global const double *y,
        global double *z )
{
    size_t idx = get_global_id(0);
    z[idx] = x[idx] + y[idx];
}
```

- The number of work-items is specified by its host code
- Single compute unit by OpenCL SDK default!
- Pipeline parallelism
- Increasing the number of work-items does not require additional FPGA resource

Geodesic Distance Kernel

- Distance between the two geographic coordinates on the earth
  - Earth's shape is modeled as an ellipsoid
  - The shortest distance is the geodesic
  - World Geodetic System (WGS) 84, which is the reference coordinate system used by the GPS

- Computing the geodesic requires higher precision
  - Double-precision is the default data type
  - Frequent calls to special functions
    - sqrt, sincos, atan2

Pseudocodes for the Geodesic Distance Kernel

```
__kernel
void geodesic_distance (   TYPE * restrict lat1,
                           TYPE * restrict lon1,
                           TYPE * restrict lat2,
                           TYPE * restrict lon2,
                           TYPE * restrict out)
{
   id = get_global_id(0) ;
   rad_lon1 = lon1[i] * TO_RADIAN ;
   rad_lat1 = lat1[i] * TO_RADIAN ;
   rad_lon2 = lon2[i] * TO_RADIAN ;
   rad_lat2 = lat2[i] * TO_RADIAN ;

   tu1 = COMPRESSION_FACTOR * sin (
   rad_lat1 ) / cos ( rad_lat1 ) ;
   tu2 = COMPRESSION_FACTOR * sin (
   rad_lat2 ) / cos ( rad_lat2 ) ;

   cu1 = 1.0 / sqrt ( tu1 * tu1 + 1.0 ) ;
   su1 = cu1 * tu1 ;
   cu2 = 1.0 / sqrt ( tu2 * tu2 + 1.0 ) ;
   s = cu1 * cu2 ;
   baz = s * tu2 ;
   faz = baz * tu1 ;
   x = rad_lon2 - rad_lon1 ;
```

Building block 0 (BB0)

```
do  {
  sx = sin ( x ) ;
  cx = cos ( x ) ;
  tu1 = cu2 * sx ;
  tu2 = baz - su1 * cu2 * cx ;
  sy = sqrt ( tu1 * tu1 + tu2 * tu2 ) ;
  cy = s * cx + faz ;
  y = atan2 ( sy, cy ) ;
  sa = s * sx / sy ;
  c2a = - sa * sa + 1.0;
  cz = faz + faz ;
  if ( c2a > 0.0 ) cz = -cz / c2a + cy ;
  e = cz * cz * 2.0 - 1.0 ;
  c = ( ( -3.0 * c2a + 4.0 ) *
    FLATTENING + 4.0 ) * c2a *
    FLATTENING / 16.0 ;
  d = x ;
  x = ( ( e * cy * c + cz ) * sy * c + y ) * sa ;
  x = ( 1.0 - c ) * x * FLATTENING +
      rad_lon2 - rad_lon1 ;
} while ( fabs ( d - x ) > EPS ) ;
```

BB1

```
   x = sqrt ( ELLIPSOIDAL * c2a + 1.0 )  + 1.0 ;
   x = ( x - 2.0 ) / x ;
   c = 1.0 - x ;
   c = ( x * x / 4.0 + 1.0 ) / c ;
   d = ( 0.375 * x * x - 1.0 ) * x ;
   x = e * cy ;
   s = 1.0 - e - e ;
   s = ( ( ( ( sy * sy * 4.0 - 3.0 ) *
      s * cz * d / 6.0 - x ) *
      d / 4.0 + cz ) * sy * d + y ) *
      c * POLAR_RADIUS ;
   out[id] = s;
}
```

BB2

Intel/Altera Arria10 FPGA

- One of the most popular FPGA chips
- The first FPGA that supports native IEEE floating-point operations in hardware
- 20-nm technology
- ~1,500 Single-Precision (SP) hardened FPUs
  - Theoretically 1.5 TFLOPS
- ~5MB of internal memory
- Hard memory controller
  - Support DDR3/4
  - Typical bandwidth 34GB/s (per board)
- PCIe accelerator or CPU-FPGA SoC
- Board vendors generally support OpenCL

Analysis of OpenCL-generated FPGA design

Number of DP operators by type and building block

Operator	BB0	BB1	BB2	Total
dp_mul	13	18	13	44
dp_div	4	2	3	9
dp_sicos	2	1	0	3
dp_atan2	0	1	0	1
dp_sqrt	2	1	1	4

Number of SP operators by type and building block

Operator	BB0	BB1	BB2	Total
sp_mul	13	16	13	42
sp_add	1	4	6	11
sp_sub	1	4	4	9
sp_multadd	2	6	3	11
sp_div	4	2	3	9
sp_sincos	2	1	0	3
sp_dot2	0	1	0	1
sp_atan2	0	1	0	1

FPGA Resource Usage and Maximum Frequency

DP

	cu1	cu1 (fpc)*	cu2	cu2 (fpc)
Logic utilization	36%	28%	61%	45%
Memory bits	14%	14%	22%	21%
RAM blocks	25%	25%	44%	38%
#DSPs	515	515	1030	1030
Fmax (MHz)	230	233	227	221

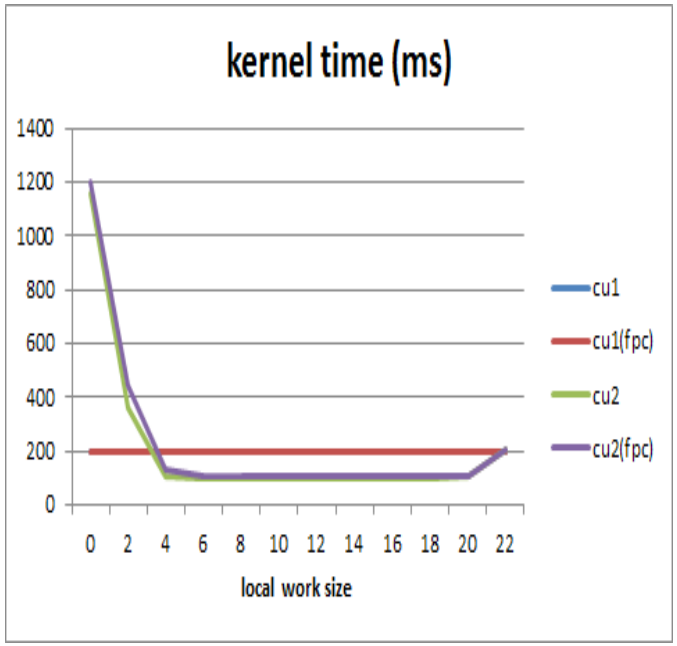
fpc: remove intermediate roundings when possible and changes the rounding mode to round towards zero for multiplies and adds

SP

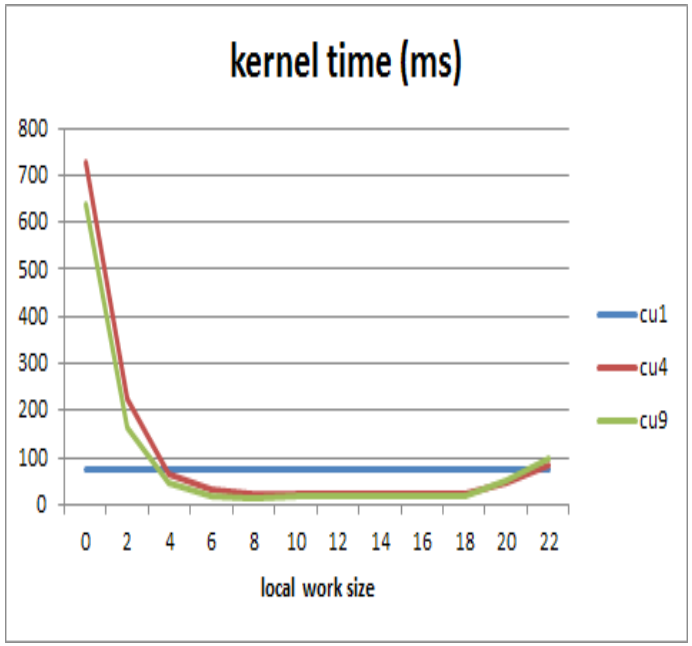
	cu1	cu4	cu9
Logic utilization	15%	28%	49%
Memory bits	8%	12%	17%
RAM blocks	18%	35%	63%
#DSPs	160	640	1440
Fmax (MHz)	280	255	212

Kernel Performance

DP

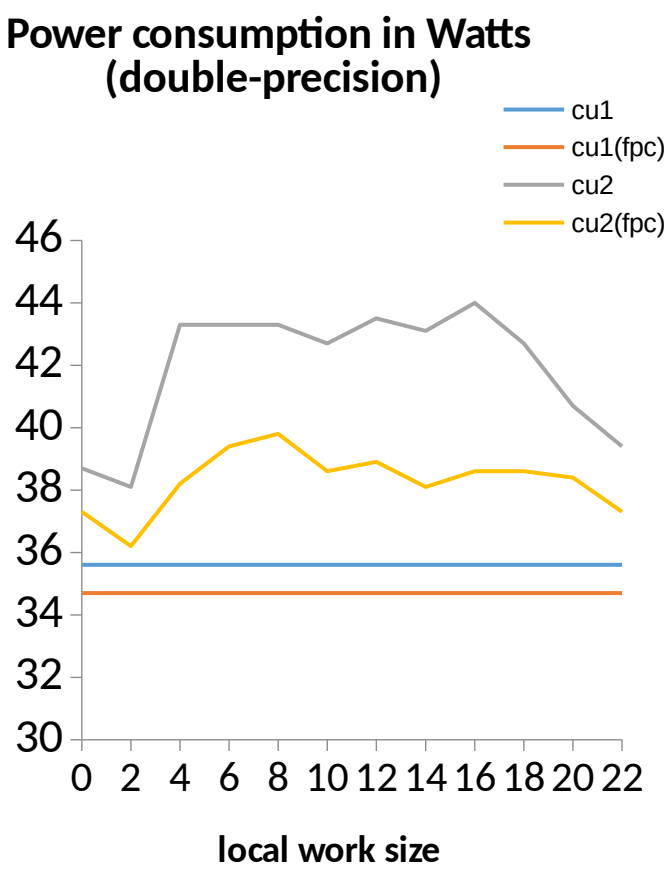


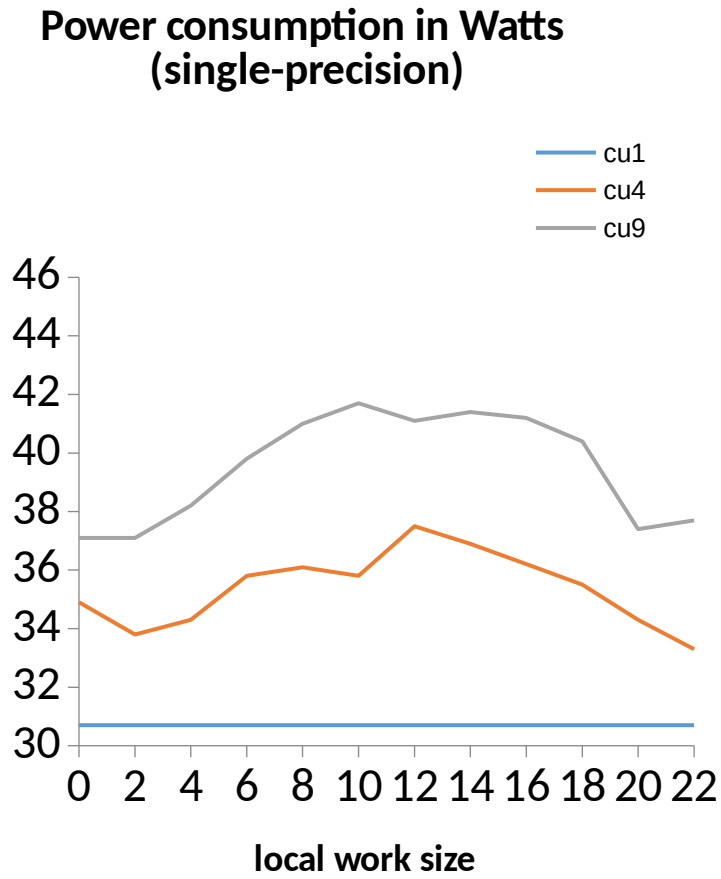
SP



Local work size = power(2, *x*-axis value)

Power Consumption

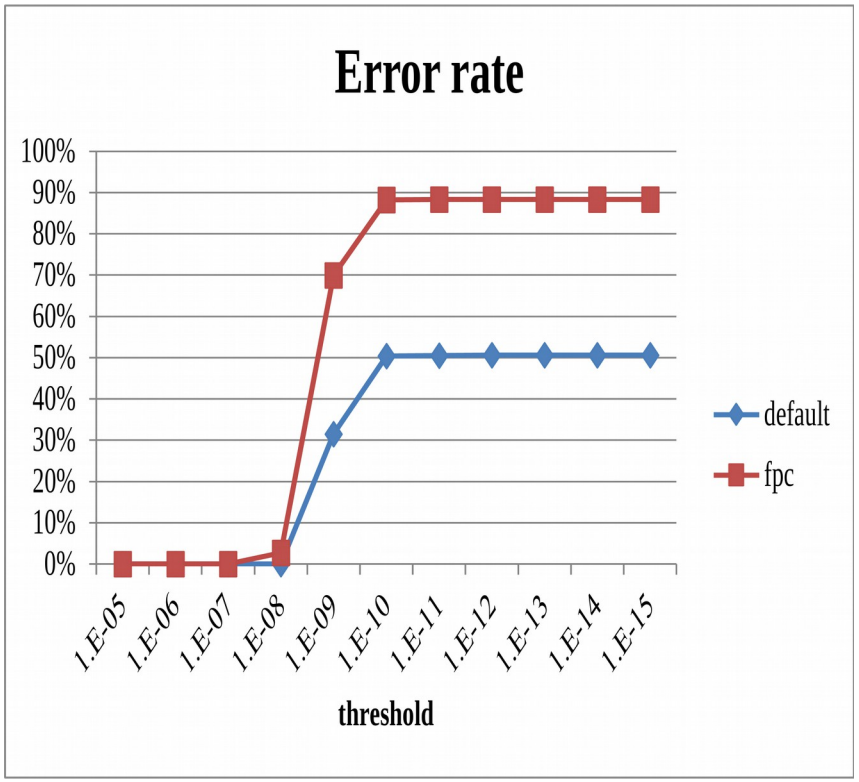




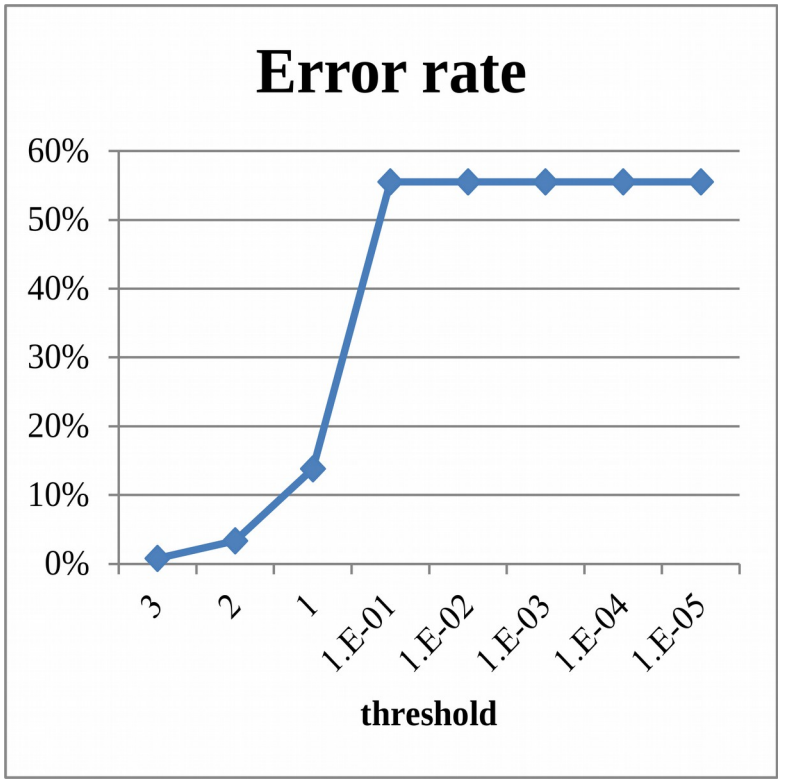
Error Rate (between FPGA and Host)

- Error rate $= \frac{1}{n} \sum_{k=1}^{n} (1 \; if \; | Rh_k - Rd_k | > err\_threshold \; 0 \; else)$

DP



SP

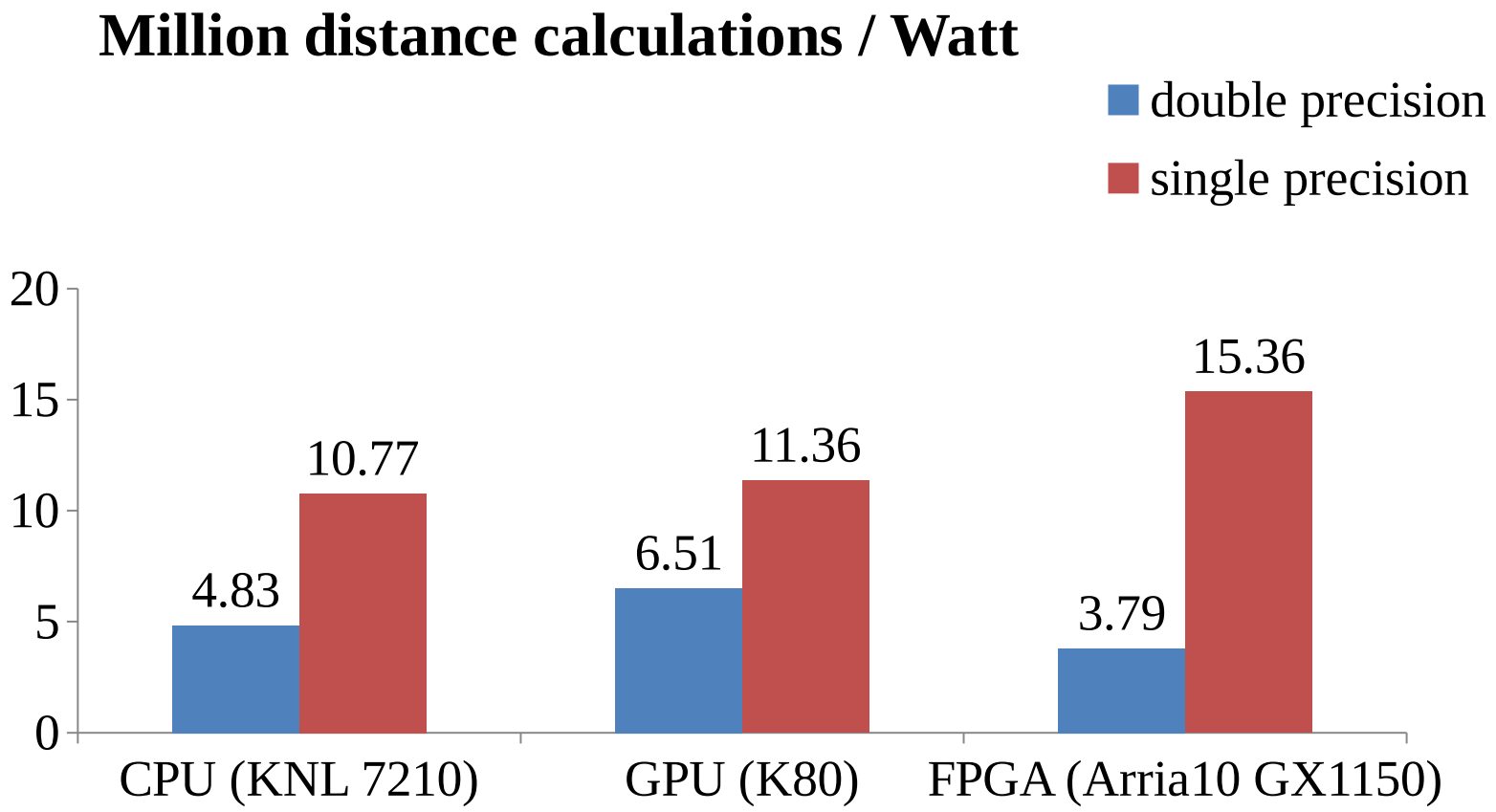


Comparison with CPU and GPU

- Intel Xeon Phi Knights Landing (KNL) 7210 CPU
  - 64 cores and 4 threads per core
  - AVX-512 SIMD enabled
- NVIDIA K80 GPU
  - 2496 cores
  - Peak DP is 2.8 TFLOPS and SP is 0.95 TFLOPS
- Nallatech 385A FPGA card
  - Intel Arria10 GX1150 device
  - Peak FP is 1.5 TFLOPS

	Time [ms]	Power [W]
CPU DP	18.3	190
CPU SP	4.0	190
GPU DP	17.7	145.5
GPU SP	5.4	136.7
FPGA DP	100.5	44
FPGA SP	13.0	42

Performance per watt on CPU, GPU and FPGA



Summary

- Single-precision (SP) kernel
  - Energy efficiency 1.42X and 1.35X better than CPU and GPU
- Double-precision (DP) kernel
  - Energy efficiency 1.36X and 1.72X worse than CPU and GPU
- Single-precision floating-point computation is suitable for the current generation of FPGA devices

Future Work

- Analyze the detailed breakdown (e.g., trig function)
- Develop techniques to accelerate the kernel
- Evaluate the kernel performance on Stratix 10
  - HyperFlex FPGA