OpenCL-enabled Parallel Raytracing for Astrophysical Application on Multiple FPGAs with Optical Links

Norihisa Fujita\textsuperscript{1,2}, Ryohei Kobayashi\textsuperscript{1,2}, Yoshiki Yamaguchi\textsuperscript{2,1}, Taisuke Boku\textsuperscript{1,2}, Kohji Yoshikawa\textsuperscript{1,3}, Makito Abe\textsuperscript{1} and Masayuki Umemura\textsuperscript{1,3}

1. Center for Computational Sciences (CCS), University of Tsukuba
2. Degree Program in Systems and Information Engineering, University of Tsukuba
3. Degree Program in Pure and Applied Sciences, University of Tsukuba
Background

• In recent years, Field Programmable Gate Array (FPGA) has been focused on High Performance Computing (HPC) area
  • FPGAs cannot defeat GPUs in performance of computation (GFLOPS) or memory bandwidth
  • We have to offload suitable works for FPGAs
• Due to improvements of FPGA tools, programming cost of FPGA is decreasing
• High Level Synthesis (HLS)
  • Application developers can make FPGA designs
• High Performance (~100Gbps) Inter-FPGA communication (xN chan.)
  • We are trying to combine computation and communication in FPGA network

<table>
<thead>
<tr>
<th></th>
<th>Performance</th>
<th>Communication</th>
<th>Programming Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>△</td>
<td>○</td>
<td>◯</td>
</tr>
<tr>
<td>GPU</td>
<td>○</td>
<td>△</td>
<td>○</td>
</tr>
<tr>
<td>FPGA</td>
<td>○</td>
<td>◯</td>
<td>× → △?</td>
</tr>
</tbody>
</table>
What is given by research

- Realizing parallel system on FPGAs using OpenCL HLS
  - Space radiative transfer code
    - Optimizing for Intel Stratix 10 FPGA
    - Parallelize on multiple FPGAs in different nodes using our inter-FPGA communication framework
  - We developed FPGA- and HLS-friendly communication system
    - Integrating networking functionality into the HLS environment
Related Work

- Performance Evaluation of HPC applications on FPGAs using OpenCL
  - Using the same code on FPGA as GPU code is not a good strategy [2]
  - OpenCL achieves the same performance as VHDL code, but consumes more FPGA resources [3]
  - Irregular memory access on FPGA has 50% more power efficiency then CPU [4]

- Communication using FPGAs
  - Microsoft makes a 6 x 8 2D-torus network for the Bing web search engine [5]
  - Noctua supercomputer at Center for Parallel Computing, Paderborn University [6]

- Communication framework for OpenCL
  - CIRCUS [14] (ours)
  - Streaming Message Interface [7] on Noctua

→ HLS for HPC and FPGA networks has been widely considered
  - However, research using both features is little known
  - In this study, we combine the productivity of HLS and high-performance network of an FPGA
Previous Work

- Implemented and optimized space radiative transfer code
  - for Intel Arria 10 FPGA [11]
  - Single FPGA only

- Inter-FPGA Comm. Framework for OpenCL
  - 40Gb Ethernet on Intel Arria 10 [a]
  - direct communication at 100Gbps on Intel Stratix 10 [14]

- In this work:
  - Combining two previous works
  - Running the code on Stratix 10 FPGAs in multiple nodes connected by optical links

ARGOT

• ARGOT (Accelerated Radiative transfer on grids using Oct-Tree)
  • Astrophysical code for the early stage of universe
  • Radiative transfer code developed in Center for Computational Sciences (CCS), University of Tsukuba
  • CPU (OpenMP) and GPU (CUDA) implementations are available
  • Inter-node parallelism is also supported using MPI
• ART (Authentic Radiation Transfer) method
  • It solves radiative transfer from light source spreading out in the space
  • Dominant computation part (90%~) of the ARGOT program on CPU
• In this research, we accelerate the ART method on an FPGA using Intel FPGA SDK for OpenCL as an HLS environment
ART Method

• ART method is based on ray-tracing method
  • 3D target space split into 3D meshes
  • Rays come from boundaries and move in straight in parallel with each other
  • Directions (angles) are given by HEALPix algorithm

• ART method computes radiative intensity on each mesh as shows as formula (1)
  • Bottleneck of this kernel is the exponential function (expf)
  • There is one expf call per frequency (ν). Number of frequency is from 1 to 6 at maximum, depending on the target problem
  • All computation uses single precision computations

• Memory access pattern for mesh data is varies depending on ray’s direction
  • Not suitable for SIMD style architecture
  • FPGAs can optimize it using custom memory access logics.

\[ I_\nu^{\text{out}}(\hat{n}) = I_\nu^{\text{in}}(\hat{n})e^{-\Delta \tau_\nu} + S_\nu(1 - e^{-\Delta \tau_\nu}) \]  

(1)
Intel FPGA SDK for OpenCL

• We used Intel FPGA SDK for OpenCL
  • Channel Extension: Transferring data between kernels directly (low latency and high bandwidth)

• Channels enable us use modularized design
  • We can use multiple kernel design to exploit space parallelism in an FPGA

• I/O Channel
  • One of channel types
  • Used for connecting between OpenCL kernels and external peripherals
  • We used it to control network controller

• BSP: Board Support Package
  • hardware component for OpenCL environment
  • contains board-specific hardware information
  • We add comm. system to the BSP
ART on Parallel FPGA

- **Processing Element (PE)**
  - OpenCL kernels to compute ART method
  - Each PE has $16^3$ sized working memory (BRAM: Block RAM)
  - Computations use data on BRAM

- **Parallelized design using channels**
  - Multiple PEs are implemented in an FPGA
    - Currently, up to 8PEs ($2 \times 2 \times 2 = 8$)
  - Channels connect between PEs to transfer ray data for radiative transfer computation
  - This structure is important for parallel computation on multiple FPGAs
Opt. for Stratix 10

- Integration on BRAM is the bottleneck
  - Although BRAM is high-speed SRAM, read-modify-write in one cycle is difficult

- In Arria 10 Implementation[12]
  - Optimized kernel written in Verilog HDL
  - However, we need to change parameters to obtain the best performance whenever we use different FPGA chips or OpenCL compilers
  - This technique is ART specific one and cannot be used for different applications.

- In this study, we implement “memory with accumulator module” in Verilog HDL
  - focus on portability for multiple applications
  - We can use this module as a “function call” in OpenCL
  - This is similar to the library on CPU

```c
ray = ray_init();
while (ray is alive) {
    pos = ray.ix * Ny * Nz + ray.iy * Nz + ray.iz;
    mesh[pos].I_nu1 += iteration
    ray->I_in_nu1 * -expm1f(-tau);
    ray = next_mesh(ray);
}
```

Pseudo-code of the behavior

CIRCUS [14]

- Intel FPGA SDK for OpenCL
  - We can describe FPGA hardware in OpenCL
- Problem: How to write inter-FPGA communication code in OpenCL?
  - MPI is the standard method for HPC applications
  - It is memory-to-memory communication, not suitable for FPGAs
  - We need to utilize pipeline-based communication in an FPGA
- → CIRCUS[14]: Communication Integrated Reconfigurable CompUting System
  - Pipelined communication and computation
    - communicate from or to a computation pipeline directly

sender(__global float* restrict x, int n) {
    for (int i = 0; i < n; i++) {
        float v = x[i];
        write_channel_intel(simple_out, v);
    }
}

receiver(__global float* restrict x, int n) {
    for (int i = 0; i < n; i++) {
        float v = read_channel_intel(simple_in);
        x[i] = v;
    }
}

CIRCUS Design Overview

- Most control logics are in OpenCL
  - Buffering, packet generation/extraction, multiplexer and demultiplexer
  - high flexibility design optimized for each application
  - Highly optimized router written in Verilog HDL in the BSP (firmware)
Router in CIRCUS

Inter-FPGA direct network (only for Albireo nodes)

64 of FPGAs on Albireo nodes (2 FPGAS/node) are connected by 8x8 2D torus network without switch

Router written in Verilog HDL and implemented in BSP → for non-neighbor FPGA comm. packet queue and switching in CIRCUS router

used in this talk
**CIRCUS performance**

max. throughput 90.2Gbps
min. latency 500ns

Throughput (1hop~7hops)  
Latency (1hop~7hops)

Evaluated on up to 8 Bittware 520N FPGA boards in Cygnus supercomputer at CCS, University of Tsukuba [14]  
**ART + CIRCUS**

- **CIRCUS** extends inter-PE channel to inter-FPGA communication

  - building a larger PE array using multiple FPGAs

---

```c
channel rt_ch[6][NPE_X][NPE_Y][NPE_Z];
channel extout_x_neg_x0_y0_z0;
channel extin_x_neg_x0_y0_z0;
#define PE_INPUT_X_POS rt_ch[3][0][0][1];
#define PE_INPUT_X_NEG rt_ch[1][0][0][0];
#define PE_INPUT_Y_POS rt_ch[5][0][0][0];
#define PE_INPUT_Y_NEG rt_ch[2][0][0][0];
#define PE_INPUT_Z_POS rt_ch[4][0][0][0];
#define PE_INPUT_Z_NEG rt_ch[6][1][0][0];
#define PE_OUTPUT_X_POS rt_ch[5][0][0][0];
#define PE_OUTPUT_X_NEG rt_ch[2][0][0][0];
#define PE_OUTPUT_Y_POS rt_ch[3][0][0][0];
#define PE_OUTPUT_Y_NEG rt_ch[1][0][0][0];
#define PE_OUTPUT_Z_POS rt_ch[4][0][0][0];
#define PE_OUTPUT_Z_NEG rt_ch[6][1][0][0];

__kernel void PE_x0,y0,z0(...) {
  while (!exit) {
    bool x_neg, x_pos, y_neg, y_pos,
      z_neg, z_pos = ...;
    if (x_neg)
      rt = read_channel_intel(PE_INPUT_X_NEG);
    else if (x_pos)
      rt = read_channel_intel(PE_INPUT_X_POS);
    else if (y_neg)
      rt = read_channel_intel(PE_INPUT_Y_NEG);
    else if (y_pos)
      rt = read_channel_intel(PE_INPUT_Y_POS);
    else if (z_neg)
      rt = read_channel_intel(PE_INPUT_Z_NEG);
    else if (z_pos)
      rt = read_channel_intel(PE_INPUT_Z_POS);
    rt = calc_intensity(...);
    else if (x_pos)
      rt = read_channel_intel(PE_INPUT_X_POS);
    else if (y_pos)
      rt = read_channel_intel(PE_INPUT_Y_POS);
    else if (z_neg)
      rt = read_channel_intel(PE_INPUT_Z_NEG);
    else if (z_pos)
      rt = read_channel_intel(PE_INPUT_Z_POS);
    calc_intensity(...);
    ...
  }
  if (x_neg_out)
    write_channel_intel(PE_OUTPUT_X_NEG, rt);
  else if (x_pos_out)
    write_channel_intel(PE_OUTPUT_X_POS, rt);
  else if (y_neg_out)
    write_channel_intel(PE_OUTPUT_Y_NEG, rt);
  else if (y_pos_out)
    write_channel_intel(PE_OUTPUT_Y_POS, rt);
  else if (z_neg_out)
    write_channel_intel(PE_OUTPUT_Z_NEG, rt);
  else if (z_pos_out)
    write_channel_intel(PE_OUTPUT_Z_POS, rt);
}
```

Pseudo-code of a PE
Limitations

• ART on FPGA
  • Can solve only the fixed size (32^3 / FPGA)
    • The size of BRAM in an FPGA limits this
      • load initial data from DDR (DDR→BRAM)
      • compute ART method (no DDR access here)
      • store the result to DDR (BRAM→DDR)
    • Future work: replacing BRAM data while computing
  • Computation does not stop if a ray crosses the intersection of meshes (l ≈ 0)

• Inter-FPGA Comm. (CIRCUS)
  • Only X- and Y-dimension support communication (for 2D-torus)
  • Flow control and error handling are not supported
  • In this paper, this is not a problem because we use up to 4 FPGAs.
  • This will be a problem when we use a greater number of FPGAs.
Evaluation Environment

• PPX
  • Pre-PACS version. X (PPX) Cluster
    • Test-bad system for next generation supercomputer in CCS, University of Tsukuba
    • Intel Xeon (BDW) CPU x 2
    • BittWare 520N (Intel Stratix10 GX 2800)
  • Direct network for FPGAs
    • 2x2 Torus @100Gbps (Opt. Cable)

• Cygnus supercomputer
  • Operation starts since Apr. 2019
  • 2x Intel Xeon CPU, 4x NVIDIA V100 GPU, and 2x Intel Stratix10 FPGA
    • 48 nodes of CPU+GPU nodes (Deneb), 32 nodes of CPU+GPU+FPGA nodes (Albireo)
    • FPGA: 2x BittWare 520N
Evaluation Problem

• Using an ART benchmark program extracted from ARGOT program
  • Input data: Uniform random numbers
  • Problem size: 32x32x32, # of Frequency(ν)=6

• Compares NVIDIA V100 (Cygnus) and BittWare 520N (PPX)
  • We must load special BSP to use CIRCUS (BSP: OpenCL HW. firmware)
    • Because Cygnus is operated for practical use, it is difficult for us to reboot the compute nodes.

• Using up to 4 nodes with weak scaling configuration
  • (1 GPU or 1 FPGA) / Node
  • weak scaling only because of FPGA’s size limitation

• Data copy time for initial data and result are excluded

<table>
<thead>
<tr>
<th></th>
<th>Cygnus (x32)</th>
<th>PPX (x4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Intel Xeon (SKL) x4</td>
<td>Intel Xeon (BDW) x2</td>
</tr>
<tr>
<td>GPU</td>
<td>NVIDIA V100 x4</td>
<td></td>
</tr>
<tr>
<td>FPGA</td>
<td>Bittware 520N x2</td>
<td>Bittware 520N x1</td>
</tr>
<tr>
<td>FPGA Network</td>
<td>100G Opt. 5/10m</td>
<td>100G Opt. 10m</td>
</tr>
</tbody>
</table>
Performance Result

• Performance
  • 4.5, 8.4 and 10.6 times faster than GPU on 1, 2 and 4 nodes

• Parallel efficiency
  • Weak scaling: computation time should be the same
  • With four nodes, FPGA is 92.4% while GPU is 49.2%
  • This comes from a low-overhead high-speed inter-FPGA communication provided by CIRCUS
Resource Utilization

<table>
<thead>
<tr>
<th>ALM</th>
<th>Registers</th>
<th>M20K</th>
<th>MLAB</th>
<th>OpenCL Freq.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>695,204</td>
<td>1,324,993</td>
<td>5,053</td>
<td>1,916</td>
</tr>
<tr>
<td></td>
<td>74.5%</td>
<td>35.5%</td>
<td>43.1%</td>
<td>33.3%</td>
</tr>
</tbody>
</table>

- Resource utilization
  - 8 PEs + CIRCUS
    - ALM is the most consumed resource
    - Difficult to implement more PEs in this FPGA
      - Other resources have space. We should optimize ALM utilization

- OpenCL frequency is little slow
  - We expect 250~300MHz on Stratix 10 devices with OpenCL
  - multiplexer in CIRCUS is the critical path
Discussion

• According to our previous study [12], $32^3$ size is too small for GPU
  • $32^3$ does not have enough parallelisms to utilize CUDA cores
    • V100 has 5120 CUDA cores
    • If we run $128^3$ size on GPU, we expect GPU will be x1.6 more effective
  • However, we suppose FPGA is still faster than GPU
    • FPGA is 4.5 times faster on a single node
    • Future work: getting rid of the limitation
  • If we do strong scaling, FPGA will be better than GPU
    • FPGA can handle small cases well

\[ y = 8x \]

(*) computation cost is $O(N^3)$

Conclusion & Future work

• We realized parallel computation using multiple FPGAs in different nodes
  • ART (Authentic Radiation Transfer) method
  • CIRCUS: Communication Integrated Reconfigurable CompUting System
  • FPGA archives up to 10.6 times faster than GPU and x3.8 performance running on 4 FPGAs

• Future Work
  • Supporting larger size on FPGA
  • Using more FPGAs on Cygnus (up to 64)
  • ARGOT with parallel FPGA with GPUs
    • FPGA-GPU DMA[b]