# Efficiency and Effectiveness Analysis of a Scratchpad Memory on FPGA and GPU for Diffuse Radiation Transfer Simulation

筑波大学 University of Tsukuba

University of Tsukuba Center for Computational Sciences 筑波大学 計算科学研究センター

FURUKAWA Kazuki YAMAGUCHI Yoshiki YOSHIKAWA Kohji KOBAYASHI Ryohei
FUJITA Norihisa BOKU Taisuke UMEMURA Masayuki

## Introduction

Radiation hydrodynamics is a fundamental scientific concept to unveil the cosmic physics process in astrophysics. In this project, we target the acceleration of a set of **Radiative Transfer (RT) simulation** code, the **ARGOT (Accelerated Radiative transfer on Grids using Oct-Tree) [1] program**. The ARGOT framework consists of two algorithms (Fig. 1), and the concrete target in this study is the **ART (Authentic Radiation Transfer)[2] scheme**, which computes the **Diffuse Radiation Transfer**. We have implemented the ART scheme on GPUs and FPGAs, and analyse them in this poster.

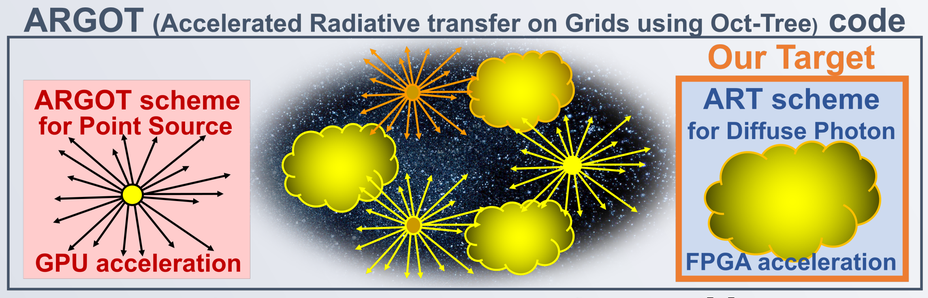


**Figure 1:** Basic Concept of ARGOT [1]

## Diffuse RT Simulation

The ART scheme takes **more than 90%** of total simulation, which is not **not matrix calculation**. It has to calculate results of the different atoms.

**Based on Ray Tracing algorithm:**

- **Rays** go straight ahead without reflection.
- Rays can have 768 or 3,072 different angles (N_ANG = 768, 3,072).
- Each Ray is computationally and geometrically **parallel and independent**.

⟶ Suitable for **many-core architectures**.

**Algorithm 1:** The most critical part of the ART

```
1:  for ipix = 0 to N_ANG−1 do
2:   for ray = 0 to N_MESH_SIDE^2-1 do
3:    I_old ← read_initial_intensity;
4:    while Ray is in the current space do
5:     M ← read_initial_mesh_data;
6:     path_length ←
7:      calc_pathlen(init_position[ray],angle[ipix]);
8:     // Radiation intensity calculation
9:     tau ← M.absorption * optical_path_length;
10:    etau ← exp(-tau);
11:    etaum1 ← 1.0 - etau;
12:    I_new ← I_old * etau + M.source_func * etaum1;
13:    // Update intensity for next mesh
14:    I_old ← I_new;
15:    // Atomic accumulation of mesh data
16:    M.accum_I ← M.accum_I + I_new * etaum1;
17:    M.accum_tau ← M.accum_tau + tau;
18:    Write_back(M.accum_I, M.accum_tau);
19:   end while // Loop for a single ray
20:  end for // Loop for rays having the same angle
21: end for // Loop for angles
```

**Memory Access is complex (Fig. 2):**

When the ART deals with $512^3$ meshes, it requires **more than 2GB** for storage.

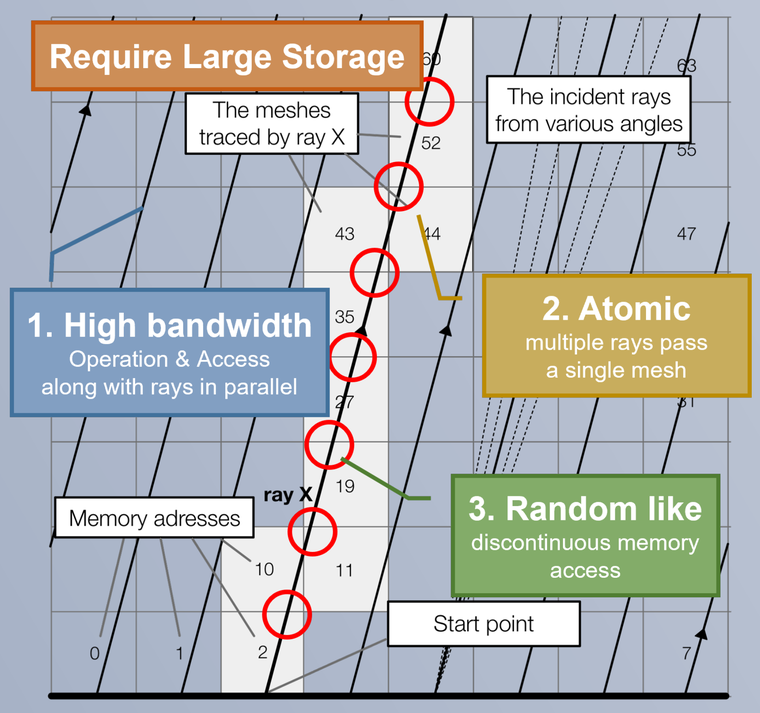


**Figure 2:** Memory Access of the ART

## Application-Specific Buffering Scheme: PRISM

**PRISM (PRefetchable and Instantly accessible Scratchpad Memory)** [3]

**Key Concept:**

1. **To increase available bandwidth**
   ⟶ Also increase memory access locality
2. **To overlap computation and memory access**

**Realised by:**

- Combination of external & internal memory
- Reusing mesh data many times on chip
- 2 of elongated prism shape subspace (Fig. 3)

⟶ To put many spaghetti (ray) bundles in it

| PRISM is | Xilinx Alveo U280 FPGA / Verilog HDL | Nvidia A100 GPU / CUDA C++ |
|---|---|---|
| made up by | **UltraRAMs** (960x36KB/Device) | **Shared Memory** (160KB/SM) |
| a size of | **72KB** (2x UltraRAM block) | **17KB** / 160KB |
| ____ for ray data | Possible to communicate with neighbours | Independent from others |
| mainly effective by | Overlapping computation and mem access | Increasing cache hit rate |

In PRISM-GPU, the parallelism is increased by dividing into the several **Ray Groups**.

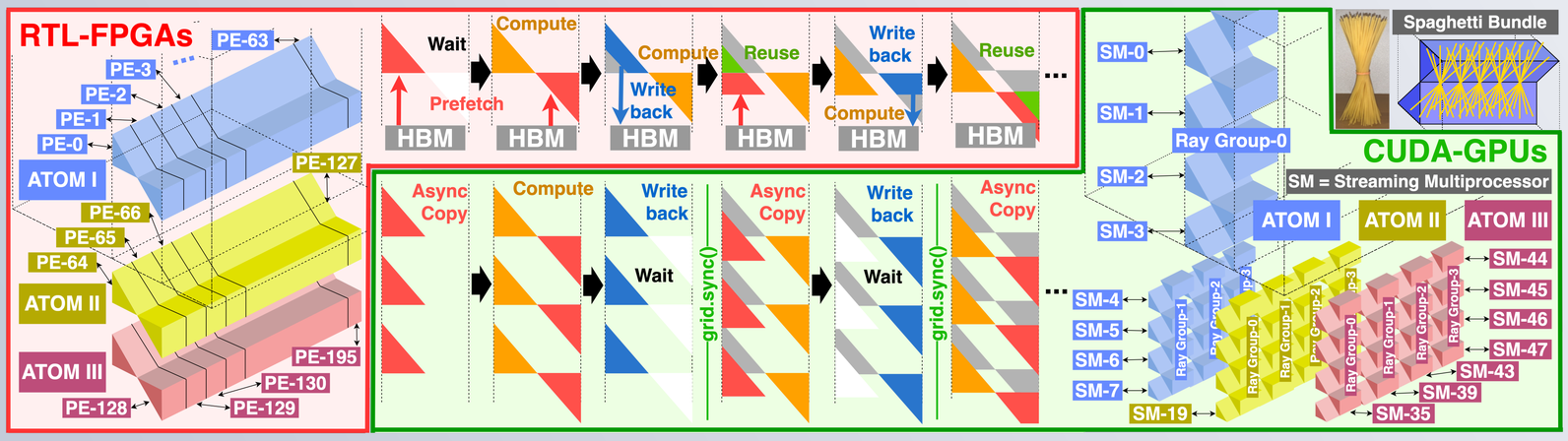


**Figure 3:** Comparison of Control Flows between PRISM-FPGA (Left) and PRISM-GPU (Right)

## Implementation Result

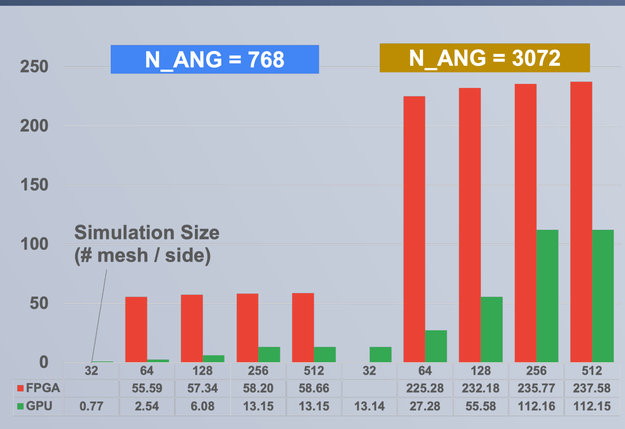


**Figure 4:** Data Reuse Rate on Scratchpad Memory (N_ANG is the number of ray angles. Data is theoretical rate. **FPGA** is always higher than **GPU**.)

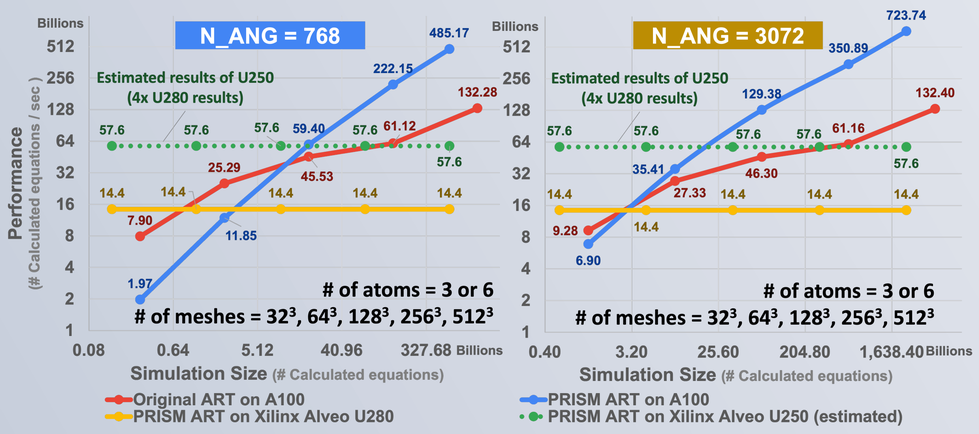


**Figure 5:** Performance Comparisons on Various Implementations (Result of **PRISM** is always the best. When the simulation size is **small**, **FPGA** is better because **GPU** cannot hide its overheads.)

## Conclusion

Using our proposed method, we conclude that the original ART can be accelerated by both FPGAs and GPUs. An FPGA yields better when the simulation space is small, while a GPU is better when large because of GPU's high parallelism. We also prove that the PRISM reduces the memory access bottleneck and contributes to a significant increase in the utilisation of the arithmetic circuits on the accelerators.

## References

[1] T. Okamoto *et al.*, "ARGOT: accelerated radiative transfer on grids using oct-tree," *Monthly Notices of the Royal Astronomical Society*, vol. 419, pp. 2855–2866, Feb. 2012.

[2] S. Tanaka *et al.*, "A new ray-tracing scheme for 3D diffuse radiation transfer on highly parallel architectures," *Publications of the Astronomical Society of Japan*, vol. 67, pp. 62(1–16), May 2015.

[3] K. Furukawa *et al.*, "An efficient RTL buffering scheme for an FPGA-accelerated simulation of diffuse radiative transfer," in *2021 International Conference on Field-Programmable Technology (ICFPT)*, pp. 1–9, 2021.

## Acknowledgements

This work was supported in part by the MEXT Next Generation High-Performance Computing Infrastructures and Applications R&D Program, entitled "Development of Computing-Communication Unified Supercomputer in Next Generation", and in part by the JSPS KAKENHI #21H04869. We would also acknowledge the support of Xilinx Inc. through the Xilinx Univ. Program.