# Extending the Hennel.jl code generation framework with multi-GPU support

1st Wiktor Łodyga
*Faculty of Electrical Engineering*
*Warsaw University of Technology*
Warsaw, Poland
wiktor.lodyga@pw.edu.pl
ORCID: 0000-0002-4531-368X

2nd Bartosz Chaber
*Faculty of Electrical Engineering*
*Warsaw University of Technology*
Warsaw, Poland
bartosz.chaber@pw.edu.pl
ORCID: 0000-0002-0917-2162

***Abstract*—Particle-in-Cell with Monte Carlo collisions (PIC/MCC) is widely used for kinetic simulations of low-pressure gas discharges. This paper presents a multi-GPU extension of Hennel.jl, a Julia framework developed by the authors that generates standalone solver modules from high-level model descriptions. The extension targets one-dimensional electrostatic PIC/MCC cases in which particle arrays dominate the data size while grid arrays are small enough to replicate on each GPU. The main contribution is to express particle decomposition as a generated transformation of the timestep: particle arrays are partitioned across GPU ranks, grid quantities are replicated, and a collective reduction of deposited charge and density arrays is inserted before the field solve while the high-level plasma model remains unchanged. The implementation was checked against Turner capacitively coupled discharge benchmark cases 1–3. In strong-scaling runs, the generated solver reaches speedups of 1.54–1.75 on two GPUs at about $1.05 \times 10^6$ macroparticles. In weak-scaling runs, the measured two-GPU throughput gain is 1.48–1.97 and reaches 1.80–1.97 near $0.52 \times 10^6$ macroparticles per GPU. A double-precision comparison with WarpX shows the same transition from overhead-dominated small cases to faster two-GPU execution in larger cases. These results show that, for small-grid 1D PIC/MCC, multi-GPU decomposition can be generated as a practical execution variant without changing the modeling interface.**

***Index Terms*—Particle-in-Cell, Monte Carlo collisions, code generation, CUDA, NCCL, multi-GPU computing, low-pressure plasma**

## I. Introduction

Particle-in-Cell with Monte Carlo collisions (PIC/MCC) is a standard kinetic method for low-pressure gas discharges. In electrostatic PIC/MCC, charged macroparticles are advanced in a self-consistent electric field, while collisions with the neutral background gas are sampled stochastically from cross-section data and particle velocities [1], [2]. A timestep combines particle-to-grid and grid operations, such as charge deposition and the solution of Poisson's equation, with operations whose cost scales mainly with the number of macroparticles.

This paper focuses on one-dimensional capacitively coupled plasma (CCP) benchmark cases. In these cases, the mesh usually contains only hundreds of cells, while the charged species are represented by many more macroparticles. The large data structures are therefore the particle arrays, whereas the grid arrays are small enough to replicate on each GPU. This makes particle decomposition a natural multi-GPU strategy for this setting: particle arrays are split between GPUs, and charge and density contributions are reduced after local deposition.

A different choice is usually made in large multidimensional PIC codes, where the grid itself is too large to replicate cheaply. Codes such as PIConGPU and WarpX are designed for electromagnetic simulations on GPU clusters [3], [4]. Their parallel execution is based on spatial domain decomposition: the mesh is divided into subdomains or patches, particles are exchanged when they cross subdomain boundaries, and field data near those boundaries are synchronized through guard cells. WarpX builds this execution model on AMReX, a block-structured adaptive mesh refinement framework [5].

Particle decomposition is an alternative to spatial decomposition for problems in which the particle population is large compared with the grid. In this variant, every rank stores the grid arrays, while the macroparticles are split between ranks. Each rank accumulates charge and density contributions from its local particles, and the partial grid arrays are then reduced before the fields are updated. Di Martino *et al.* used this idea in a parallel PIC plasma code [6]. Related replicated-grid variants also appear in hybrid schemes; for example, Hatzky described domain cloning for PIC on clusters of symmetric multiprocessor computers [7]. Qiang and Li compared particle-field decomposition with domain decomposition in parallel PIC beam dynamics and showed that the preferred decomposition depends on the balance between macroparticle count, grid size, and processor count [8]. In this sense, particle decomposition is not new; what matters here is how it can be expressed inside a generator.

Hennel.jl is a research framework, developed by the authors and written in Julia [9], for generating standalone solver modules from high-level model descriptions. The model is described once, and the backend emits code for the selected numerical method and hardware target. This changes the role of the decomposition: instead of being written into a separate solver by hand, it can be introduced while the timestep is generated. Because the generator has access to the structure of the timestep, it can distinguish stages that operate on rank-local particles from stages that require a globally consistent grid state.