A GPU BASED 3D PARTICLE TRACKING CODE FOR MULTIPACTING SIMULATION*

T. Xin#,1,2, I. Ben-Zvi1,2, S. Belomestnykh1,2, J. C. Brutus1, V. N. Litvinenko1,2,
I. Pinayev1, J. Skaritka1, Q. Wu1, B. Xiao1
1Brookhaven National Laboratory, Upton NY 11973, USA
2Stony Brook University, Stony Brook NY, 11794, USA

Abstract

A new GPU based 3D electron tracking code is developed at BNL and benchmarked with both popular existing parallel tracking code and experimental results. The code takes advantage of massive concurrency of GPU cards to track electrons under RF field in 3D Tetrahedron meshed structures. Approximately ten times more FLOPS can be achieved by utilizing GPUs compare to CPUs with same level of power consumption. Different boundary materials can be specified and the 3D EM field can be imported from the result of Omega3P calculation. CUDA_OpenGL interop was implemented so that the emerging of multipactors can be monitored in real time while the simulation is undergoing. Code also has GPU farm version that can run on multiple GPUs to further increase the turnover of multipacting simulation.

INTRODUCTION

Electron multipacting (MP) study in an SRF cavity and power coupler is of great importance in both designing and operating phase of the device. There are several 2D codes that can handle structures with cylindrical symmetry such as Multipac and Fishpact. To deal with 3D structures we have Track3P solver in the ACE3P package and Particle Studio in the CST suite. For 2D codes the limitation is obvious, especially when we are facing a power coupler problem where the structures are usually lack azimuthal symmetry. The Track3P code is extremely powerful in terms of the range of problems it can handle but it also requires a cluster such as NERSC to fully harness this power. Therefore we developed this GPU based 3D tracking code to increase the turnover of the multipacting simulation in SRF structures with only several GPU cards. This code can run on either PC or workstation as long as a GPU that support Nvidia CUDA computing capability 1.3 and above is available.

STRUCTURE OF THE CODE

The idea of this code is to take the advantage of high concurrency of the GPU to run a large scale Monte Carlo process to simulate the multipacting phenomenon. There are three primary parts in the code.

Main (Master) Function

The main function is a host function that runs on CPU and controls the work flow of the program. All the kernels running on GPU are launched from the main host code. First, the input parameters are read into the main function from an input file. Then the geometry model of an RF structure and the field distribution from Omega3P eignesolver are read in. The mesh model will be pre-processed before it is sent to the GPU so that the particles can be more easily located when it is going through the tracking process. Then the main function calls the sequence of the core kernels in the display call back function of the OpenGL so that the tracking process is synchronized with the rendering process. The core tracking kernels will be discussed below.

Momentum Update

Initial locations, momentums and relative RF phases of the particles are generated by a kernel called init_par on GPU. Then the field strength at the location of the particle is calculated by using first order shape function of the Tetrahedral element and the field info on the vertex of the element in which the particle is located. After the field information is ready, the momentum updating kernel takes the pointer to the chunk of global memory that stores the information and starts calculating the new momentum of the particles. The momentum update task was done by a fourth order Runge-Kutta method. After the momentum is updated, the new location of the particle is also updated as if there is no collision. We call this a virtual movement of the particle. The further steps will be conducted by the particle locating kernel, which will be discussed in the following section.

Since momentum updating does not involve the memory copying between GPU and CPU, it can save us considerable amount of time. Each momentum updating kernel requires 90 registers, which still needs some optimization. But even with this amount of register requirement a low tier GeForce GTX 860M GPU can run about 50 thousands of the kernels simultaneously and do one iteration in about 30 ms for 2 million particles.

Particle Locating

The code spends major portion of time on locating the particle after each virtual movement of the particle. The data structure of the mesh model is organized so that each mesh element also stores the ID of the four neighbor elements as well as the internal ID of the surfaces they shared. Every time after the particle is virtually moved by

* This work was carried out at Brookhaven Science Associates, LLC under Contracts No. DE-AC02-98CH10886 and at Stony Brook University under grant DE-SC0005713 with the U.S. DOE.

#txin@bnl.gov

the momentum updating kernel, we follow the procedure of following pseudocode [1]:

```
Calculate the Barycentric coordinates (B-coords) of the particle in the old element.
If the particle is still in the old tet mesh (all B-coords > 0 ):
    Make a real movement on the particle;
    Update the momentum of the particle;
    Return;
Else:
    Calculate the parametric coordinates (a, b) of intersect of the virtual trajectory of the particle with the walls of the tet element; there will be one and only one pair of coordinates falls in the range (0<a<1; 0<b<1; a+b<1) and the side corresponding to that point is the side that the virtual trajectory went through.
    If the particle hit a wall for the first time:
        Record the ID of the wall hit by the particle;
        Generate the new momentum of the particle as if it is a new particle;
        Flag the particle as a particle that just hit a wall;
        Return;
    Else if the particle hit the wall in previous time step:
        Flag the particle as dead;
        Return;
    Else if the particle hit a shared wall:
        Update the ID of tet mesh the particle is located and go back to the top of the code.
```

The code actually keeps a record of how many steps the kernel took to eventually find the particle and adjust the interval of time step of each individual particle accordingly. If the kernel took too long to locate the particle then the time step will be halved in next iteration.

After every particle is either located or registered dead, the master clock advance by one time step and the EM field is updated.

After every certain amount of time, 2 RF cycles is set as a default for now, the master function calls the memory copy of the results back from GPU to CPU and does a sort on the flags to get rid of the dead particles so that the following simulation can focus on the active particles only.

The particle locating kernel requires significantly more registers (180) and has much more branching than the momentum update kernel. Both are undesirable for a GPU code. However the maximum achievable GFLOPs of this kernel on a Tesla K40 card is still around 150 which is about three times than an Intel i7 quad-core CPU can provide. Although there is still ample of space for improvement on the algorithm of this kernel, we already can see the power of a GPU accelerated code.

PERFORMANCE AND RESULT BENCHMARKING

The performance of GPU kernels and of an equivalent CPU code is shown in Figure 1.

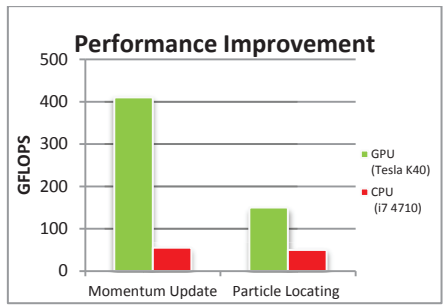


Figure 1: Performance improvement by implementing GPU kernels.

As we can see, the code runs about three times as fast on an Nvidia Tesla k40 GPGPU card than on an Intel i7 quad-core and the work can be easily distributed to multiple GPU cards on a GPU farm due to the independency between the different jobs.

We did three set of benchmarking between our GPU code and the Track3P code as well as with the experimental results. In most of cases we saw certain level of consistency between the simulation results and the experiment observations.

112 MHz QWR Injector

The 112MHz QWR cavity has a coaxial fundamental power coupler and a choke joint structure cathode stalk. Figure 2 shows the section view of the cavity [2].

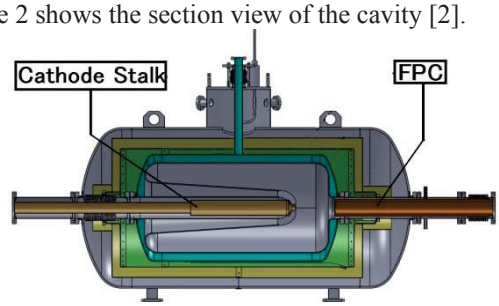


Figure 2: Section view of the 112MHz cavity.

The simulation results from GPU code and Track3P are shown in Figure 3 below. We can see consistency between the two results. There are three main multipacting bands. The first one appears when the gun voltage reaches 40 to 50 kV. This MP band is located inside the cavity. The second one emerges at around 200 kV gun voltage and is persistent to about 650 kV. This MP is located in FPC structure. The third one is located in the cathode stalk and the corresponding gun voltage is from 600 kV to 1 MV. The locations of the MP bands are shown in Figure 4.

ISBN 978-3-95450-178-6

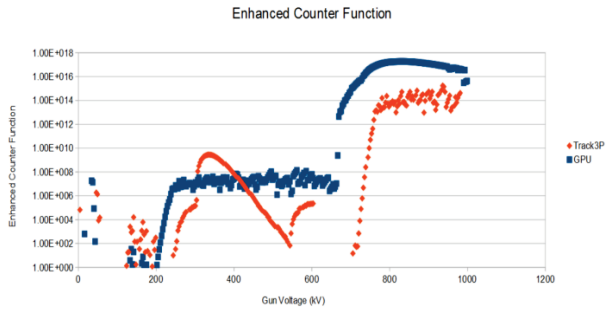
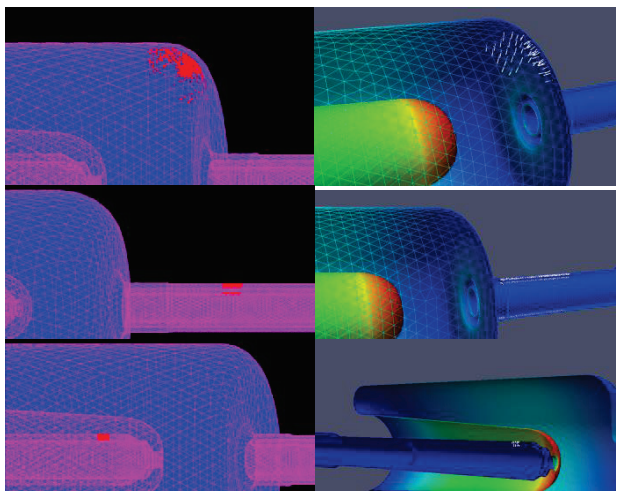


Figure 3: Simulation results of the enhanced counter function for the 112 MHz cavity by GPU code and Track3P.



(a) (b)

Figure 4: Location of MP in the 112 MHz cavity given by (a) GPU code shown as red dots and (b) Track3P shown as white dots. The pair of pictures from top to bottom shows the MP at gun voltage equal to 40 kV, 200 kV and 600 kV respectively.

All the MP bands were observed in the gun commissioning at BNL early this year. Fortunately we were able to overcome MP eventually [3].

2.1 GHz 3-Cell Cavity with Coupling Slots

The 2.1 GHz normal conducting 3-cell cavity was designed for the Low Energy RHIC electron Cooler (LEReC) as an energy spread correcting cavity. The original design has coupling slots between the neighbouring cells. The multipacting study was needed at the time and we decided to use this as a benchmark for the new GPU code. The simulation results of the enhanced counter function are shown in Figure 5 and the locations of the MP are shown in Figure 6.

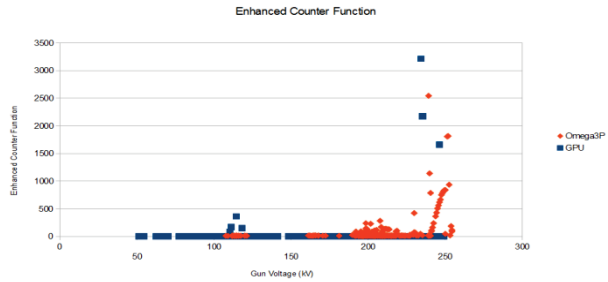


Figure 5: Enhanced counter function for the 2.1 GHz cavity

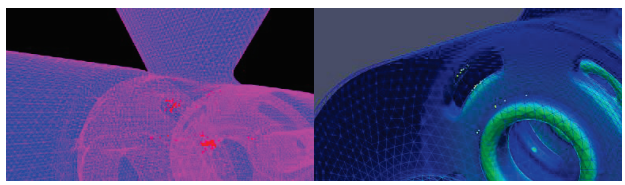


Figure 6: Location of MP in the 2.1 GHz cavity at Vg = 240 kV

As we can see, MP at 240 kV is primarily located in the coupling slots. And both code give similar predictions.

56 MHz QWR Cavity

The commissioning of the 56 MHz SRF QWR cavity has also been done early this year at BNL. This gives us an opportunity to cross compare observations of MP behavior of this cavity, the previous simulation results done by Track3P and the simulations by our GPU code. The simulation results of enhanced counter function and location of MP are shown in Figures 7 and 8 respectively.

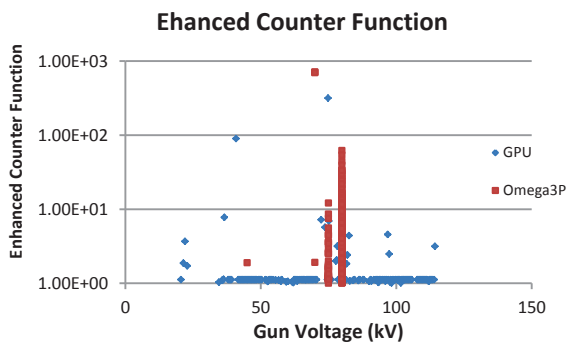


Figure 7: Enhanced counter function of 56 MHz cavity

The experimental observation shows that at around 50 kV cavity voltage there was some relatively strong MP [4]. This can be seen as the confirmation of the simulation result from both GPU code and Track3P.

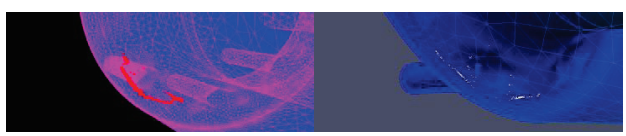


Figure 8: Location of MP in 56 MHz cavity at Vg=40 kV. The MP is located at the connection between the FPC port and the cavity back wall.

ISBN 978-3-95450-178-6

CONCLUSION

We developed a GPU-based 3D particle tracking code for MP simulation. The code gives comparable results to Track3P and agrees with the experimental observations to a certain level. The code can run on multiple Nvidia GPU cards simultaneously and the performance scales near linearly with the number of devices.

ACKNOWLEDGMENT

The authors wish to thank Dr. Li from SLAC for his help with Track3P issues.

This research used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

REFERENCES

[1] A. Haselbacher, F.M. Najjar, J.P. Ferry, Journal of Computational Physics **225**(2), 2198 (2007).

[2] T. Xin et al., Simulations of Multipacting in the Cathode Stalk and FPC of 112MHz Superconducting Electron Gun, TUPPD082, Proc. of IPAC2012, New Orleans, Louisiana, USA (2012).

[3] S. Belomestnykh et al., Commissioning of the 112 MHz SRF Gun, THPB058, Proceedings of SRF 2015, Whistler, Canada.

[4] Q. Wu et al., Beam Commissioning of the 56 MHz QW Cavity in RHIC, WEBA07, Proceedings of SRF 2015, Whistler, Canada.

ISBN 978-3-95450-178-6