HPC - 3. Assignment - 2D Gray-Scott Model in CUDA

Kristjan Kostanjšek, Nejc Ločičnik

I. Introduction

The Gray-Scott model is a well-known example of a reaction-diffusion system, which describes how the concentration of chemical substances changes over space and time due to local reactions and diffusion. In this report, we implement a simulation of the Gray-Scott model to observe the formation of complex patterns over a two-dimensional grid, example of such pattern can be observed on Figure 1. We first developed a sequential version of the program to establish a correct and reliable baseline. Afterwards, we optimized the program using CUDA for parallel execution on a GPU, exploring additional techniques such as leveraging shared memory and multi-GPU setups. Finally, we measured and analyzed the performance improvements achieved by these optimizations, comparing the speedups of the parallelized implementations against the original sequential version. This report presents the implementation details, results, and analysis of our findings.

Figure 1. Example of a pattern generated by our Gray-Scott model.

II. Optimizations and Results

In this section we will present the results of the baseline program and the optimized versions, compare their execution times and analyze them.

A. Sequential Version

First we created the sequential version of the program. We ran the program on 5 different grid sizes, each run was limited to 5000 simulation steps. The execution times (in seconds) depending on the grid size are presented in the Table I. The measured execution time includes only the Gray-Scott simulation itself, with grid initialization excluded from the timing.

grid size	256x256	512x512	1024x1024	2048x2048	4096x4096
$t_S[s]$	13.08	46.56	182.36	721.63	2882.33

Table I
Execution times for sequential program.

B. Basic Parallelization

Parallelizing the Gray-Scott simulation is straightforward using CUDA, as the computation of each cell in the grid is independent of the others within a single simulation step. This means that for every step of the simulation, each cell can be assigned to its own GPU thread, allowing the entire grid to be processed in parallel. This basic parallelization approach alone provides a significant speedup compared to the sequential implementation, especially as the grid size increases. The achieved execution times and speedups for different grid sizes are presented in Table II. As before, the measured execution times include only the core simulation phase, excluding both the initialization of the grid and the data transfers between the host and device.

grid size	256x256	512x512	1024x1024	2048x2048	4096x4096
$t_P[s]$	0.032	0.066	0.159	0.462	1.828
speedup	408.79	705.42	1146.90	1561.96	1576.77

Table II
Execution times and speedups for basic parallel program.

C. Shared Memory Utilization

TODO: Kako in kaj, kkšna tabela za to za kater block size se dobi kakšne speedupe. Kaj je optimal block size?

D. Mulitple GPUs Utilization

TODO: Kako in kaj, kakšni so speedupi, pokomentiramo rezultate.

E. Final Results

TODO: Final, best results, speedup če združimo Shared Memory in Multiple GPUs.

III. Conclusion

TODO.