

BRINGING “GAME OF LIFE” TO “LIFE” WITH CUDA PARALLEL COMPUTING PLATFORM

Vinnitsia National Technical University

Анотація

У роботі наведено ретельний аналіз технічних аспектів впровадження програмного забезпечення, практичного застосування та обґрунтування подальшої розробки програмних продуктів на основі технології CUDA для паралельних обчислень на графічних відеокартах NVIDIA. Дослідження базується на прискоренні обчислень, що виконуються ітеративним алгоритмом для моделювання клітинного автомату, відомого як «Гра життя». Особлива увага приділяється перенесенню обчислювально-інтенсивних процесів з центрального процесора на графічний процесор NVIDIA, а також досягненню максимальної продуктивності за рахунок оптимального використання обох. У висновку підкреслюється роль паралельних обчислень у прискоренні обчислювально-інтенсивних програм.

Ключові слова: паралельні обчислення; CUDA; клітинний автомат; симуляція; покоління; алгоритм; графічний процесор (ГП); центральний процесор (ЦП); «Гра життя».

Abstract

The work provides a thorough analysis of the technical aspects of software implementation, practical applications, and justification for further development of software products based on CUDA technology for parallel computing on NVIDIA graphics cards. The research is grounded in accelerating computations performed by the iterative algorithm to simulate the cellular automaton known as the "Game of Life." Special attention is given to shifting compute-intensive processes from the central processor to the NVIDIA graphics processor, as well as achieving maximum performance through optimal utilization of both. In conclusion, the role of parallel computing in speeding up computationally intensive programs is highlighted.

Keywords: parallel computing; CUDA; cellular automaton; simulation; generation; algorithm; graphics processor (GPU); central processor (CPU); “Game of Life” (GoL).

Introduction

The rapid pace of technological advancement presents significant challenges for organizations seeking to maintain efficiency. To address these demands, many industries have developed tools that accelerate and streamline workflow processes. One such tool is CUDA, or Compute Unified Device Architecture, a parallel computing platform and programming model developed by NVIDIA. CUDA enables computational workloads to leverage the processing power of GPUs, regardless of the graphics API in use [1]. This platform plays a vital role in addressing real-world challenges, including generative artificial intelligence applications such as diffusion models and large language models (LLMs), medical analysis simulations, and neural network training in machine learning problems [2].

Research results

The cellular automaton known as the “Game of Life” (GoL) is an algorithm that benefits substantially from parallel processing. Invented by John Conway in 1970, the classical GoL is also referred to as “Conway’s Game of Life.” In this model, each cell within an infinite grid exists in one of two states: “alive” or “dead.” The state of each cell is updated at discrete time steps, with each generation defined by the collective states of all cells at a given step. The subsequent state of a cell is determined by counting the number of “alive” cells among its eight neighbouring cells.

The rules of GoL are as follows:

1. If a cell is “dead” and it has exactly 3 “alive” neighbours, then it becomes “alive”.
2. If a cell is “alive” and is surrounded by 2 or 3 “alive” cells, it remains “alive”; otherwise, it dies from “overpopulation” or “underpopulation”.

Although a simple iterative approach is intuitive and easy to implement, it is generally inefficient for large-scale grids or scenarios with repeated patterns. As a result, more advanced methods such as HashLife or QuickLife are often preferred for these applications. This consideration prompts the question of whether the classical iterative method could remain viable if its performance were significantly enhanced through optimization.

The structure of Conway’s Game of Life is inherently parallel. Because the state of each cell in generation $t+1$ depends solely on the states of its Moore neighborhood in generation t , there are no data dependencies between cells during the same update cycle. This property enables a direct mapping in which each cell in the grid is assigned to an individual GPU thread. In a CUDA-based implementation, the grid is partitioned into blocks of threads. To ensure that every cell is processed, the number of blocks must be dynamically calculated based on the total number of cells N and the fixed thread block size (commonly 256 threads for optimal occupancy). This relationship is defined by the following formula:

$$numBlocks = \left\lceil \frac{N}{256} \right\rceil \quad (1)$$

The implementation utilizes the ceiling function to ensure exhaustive coverage across the computational domain, specifically addressing scenarios where grid dimensions are not perfect multiples of the thread block size. By partitioning the workload accordingly, the GPU can schedule thousands of threads concurrently.

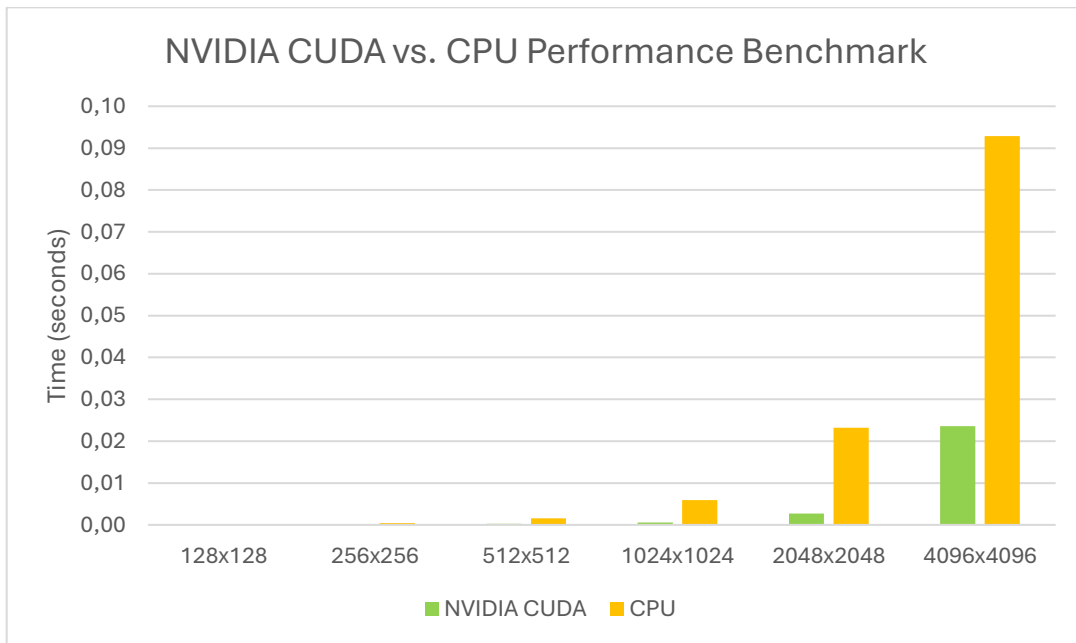


Figure 1. Comparison of execution time per generation between sequential CPU and parallel CUDA implementations across varying grid resolutions.

The benchmark data confirms that while the classical iterative method is computationally expensive for a CPU, it remains highly viable when offloaded to a parallel architecture. The GPU’s ability to handle over 16 million cells (4096^2) in a fraction of the time required by the CPU demonstrates that architectural optimization can substitute for complex algorithmic shifts (like HashLife) in scenarios where real-time visualization or raw iterative accuracy is required. Ultimately, for large-scale simulations, the CUDA implementation provides a speedup of roughly four times at peak resolution, demonstrating that the bottleneck in Conway’s Game of Life is not the algorithm’s logic, but rather the hardware’s sequential execution limits.

Conclusion

The implementation of a parallelized iterative algorithm for Conway's Game of Life (GoL) illustrates the effectiveness of the NVIDIA CUDA platform in optimizing computationally intensive grid-based simulations. Utilizing extensive parallelism allows the approach to address the sequential bottlenecks inherent in CPU-based methods. While architectural limitations, including global memory latency and thread synchronization overhead, persist, the CUDA framework remains a principal methodology for maximizing hardware throughput and achieving significant computational speedups.

REFERENCES

1. CUDA Programming Guide. URL: <https://docs.nvidia.com/cuda/cuda-programming-guide/index.html> (дата звернення: 25.01.2026).
2. CUDA. URL: https://en.wikipedia.org/wiki/CUDA#Usages_of_CUDA_architecture (дата звернення: 25.01.2026).

Щуцький Павло Сергійович – студент групи 2ПІ-24Б, факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, м. Вінниця, e-mail: pavlo050706@gmail.com.

Науковий керівник: **Чопляк Вікторія Володимирівна** – викладач англійської мови, кафедра іноземних мов, Вінницький національний технічний університет, м. Вінниця, e-mail: nikavnuchkova@gmail.com.

Pavlo S. Shchutskyi – student of group 2PI-24B, Faculty of Information Technologies and Computer Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: pavlo050706@gmail.com.

Scientific supervisor: **Victoria V. Chopliak** – teacher of English, Foreign Languages Department, Vinnytsia National Technical University, Vinnytsia, e-mail: nikavnuchkova@gmail.com.