Report: CUDA Programming Project

Introduction

Conway's Game of Life is a game devised by the British mathematician John Horton Conway in 1970. It is a zero-player game, meaning that its evolution is determined by its initial state. Each cell in the game can be in one of two states, they are alive or dead. The game follows a set of rules that determine whether a cell will live, die, or multiply in the next generation based on its surrounding cells.

The objective of this project is to implement Conway's Game of Life using CUDA, a parallel computing platform and application programming interface model created by NVIDIA. CUDA allows developers to use the power of NVIDIA GPUs for processing, making it ideal for computationally intensive tasks.

Result

The graph below comparing the time taken for each generation of Conway's Game of Life on the CPU, GPU, and GPU with shared memory illustrates the significant performance benefits of using GPU acceleration. The computations were run with a board size of 32,768 x 32,768 matrix. The CPU implementation requires around 20,000 milliseconds to compute each generation, while the GPU implementations achieve this in mere hundreds of milliseconds. Specifically, the GPU without shared memory completes each generation in approximately 346.2 milliseconds on average, showing a substantial improvement over the CPU. Moreover, the GPU with shared memory offers a slight enhancement, with an average generation time of 335.5 milliseconds, demonstrating the effectiveness of shared memory in optimizing memory access and computation.

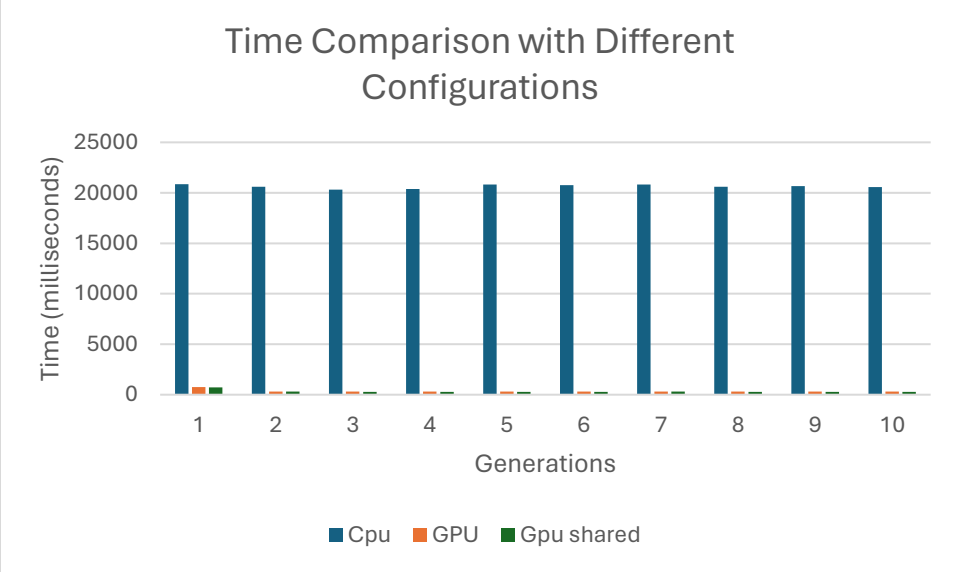

Figure 1 Time Comparison with Different Configurations

The next graph presents the average time, in milliseconds, taken to generate new generations of Conway's Game of Life for many board sizes. As the board size increases from 2048 to 32768 cells, the CPU implementation shows a substantial increase in processing time, from 74 milliseconds to 20640 milliseconds. In contrast, the GPU implementation completes the same task in only 1.2 milliseconds for 2048 cells and 346.2 milliseconds for 32768 cells. Similarly, the GPU with shared memory demonstrates a notable improvement, with times of 1.2 milliseconds for 2048 cells and 335.5 milliseconds for 32768 cells.

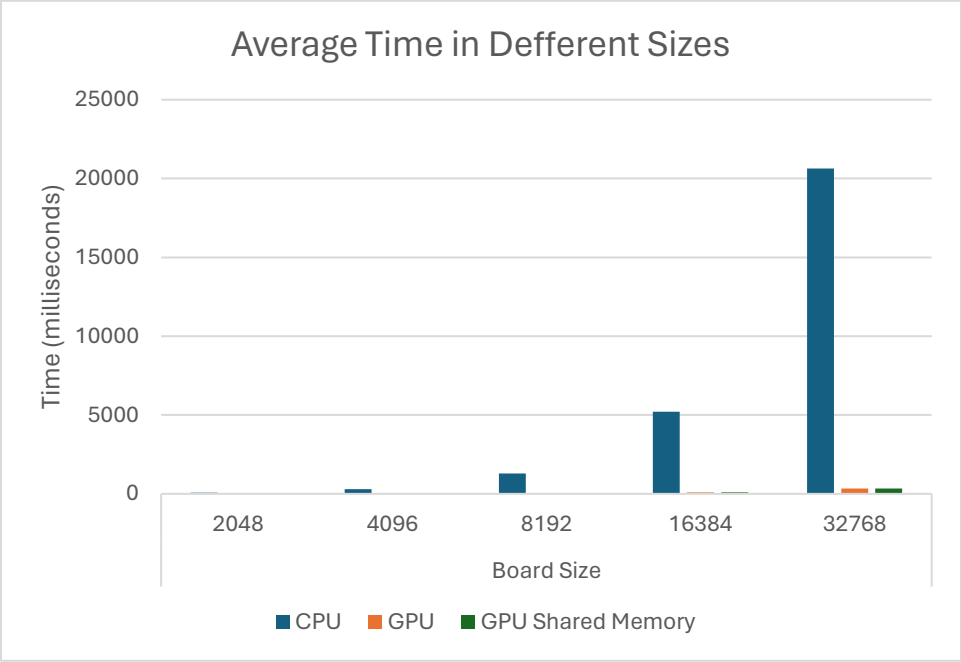

Figure 2 Average Time in Defferent Sizes

Terminologies

OpenCl	CUDA	Description
Platform	Platform	Represents the hardware and software environment for parallel computing.
Device	Device	Refers to the GPU or other processing unit used for parallel computation.
Processing Element	Core	Component that executes instructions.
Compute Unit	Multiprocessor	Unit that executes kernels.
Context	Context	Manages resources and controls execution on a device.
Kernel	Kernel	Function that runs on a device and performs computation.
Event	Event	Used for synchronization and tracking tasks.
Buffer	Memory Object	Linear memory region for data storage.
Image	Texture Object	Data structure for representing image data.
Command Queue	Stream	Manages commands and data transfers between the host and device.
NDRange	Grid	Range of work items in a kernel.
	Block	
Work Item	Thread	Unit of execution in a kernel, representing a thread or processing element.
Work Group	Thread Block	Group of work items that can share data and synchronize execution.
Global Memory	Global Memory	Memory accessible to all work items, used for shared data.
Local Memory	Shared Memory	Memory shared among work items within a work group.
Constant Memory	Constant Memory	Read-only memory accessible to all work items.
Private Memory	Registers	Memory accessible to a single work item.
Barrier	__syncthreads()	Synchronizes work items or threads.
Local Size	Block Size	Number of work items in a work group.
Global ID	Thread ID	Unique identifier for a work item.