**Accelerating Electromagnetic Simulations: GPU Implementation of FDTD Algorithm**

|  |  |  |
| --- | --- | --- |
| Zulfiqar Mohammadi Student a1850245@student.adelaide.edu.au | Thomas Woolford Industry supervisor thomas.woolford@swordfish.com.au | Olaf Maennel Academic supervisor olaf.maennel@adelaide.edu.au |

ABSTRACT

Radar propagation models are essential in many applications, yet their computational demands often result in extended simulation times. This research explores leveraging Graphics Processing Units (GPUs) to enhance the performance of radar propagation simulations by implementing the Finite-Difference Time-Domain (FDTD) algorithm on GPUs. The FDTD method, widely used for solving Maxwell’s Equations, requires significant computational power, especially for large spatial domains or long non-sinusoidal waveforms. By harnessing the data parallelism of modern multicore architectures like GPUs, this study aims to accelerate electromagnetic simulations and reduce computation times.

This research includes OpenMP implementations of the parallel FDTD algorithm, evaluating their performance on various GPUs in terms of speed and scalability. It investigates the performance and accuracy trade-offs of different propagation models, assessing their suitability for GPU implementation. Source codes for these implementations are provided to facilitate reproducibility and further exploration. Ultimately, this work quantifies the performance benefits of GPU acceleration, offering insights into the practical advantages of using GPUs for computationally intensive radar propagation simulations.

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; • **Networks** → Network reliability

1 INTRODUCTION AND MOTIVATION

In recent years, Graphics Processing Units (GPUs) have gained prominence as powerful accelerators for computationally intensive tasks due to their exceptional computational ability. Their utilization in general-purpose programming has transformed various fields, including bioinformatics, computational physics, engineering simulations, and image processing (Ying Xu & Rongling Jiang 2016). One such demanding application poised to benefit from GPU acceleration is the simulation of electromagnetic (EM) fields using the Finite-Difference Time-Domain (FDTD) method. This numerical scheme is widely employed for EM field simulation n but demands substantial computational resources and fast memory access to accurately model real-world scenarios (Demir, V 2014). This study focuses on the collaboration between Central Processing Units (CPUs) and GPUs to develop a parallel algorithm for the FDTD method. Leveraging the OpenMP API for GPU implementation, the aim is to exploit the parallel processing capabilities of both CPU and GPU architectures to accelerate EM field simulations.

The motivation for investigating CPU-GPU collaboration in parallelizing the FDTD method stems from the urgent need to address the computational demands of EM field simulations. These simulations are crucial for various applications, including radar systems and wireless communication networks. However, the complexity of real-world EM scenarios often requires significant computational resources and entails substantial computational overheads. By harnessing GPUs' parallel processing capabilities alongside CPUs' multi-core capabilities through the OpenMP API, this research aims to accelerate EM field simulations. This effort aligns with the broader objective of advancing computational techniques to meet the escalating demands of EM modeling across various domains, fostering innovation and enabling the development of more efficient and scalable simulation methodologies.

2 LITERATURE REVIEW

2.1 Project Scope

As discussed above, this project aims to enhance the speed and precision of electromagnetic simulations by utilizing both CPUs and GPUs collaboratively.

2.1.1 Optimizing FDTD Algorithm on GPU Clusters:

In this phase of research, the aim is to maximize the utilization of GPUs for accelerating electromagnetic simulations. Given GPUs' proficiency in parallel processing, our aim is to fully harness this capability. Advanced techniques will be implemented such as utilizing GPU texture memory and pinned host memory to enhance data access speed (Ying Xu & Rongling Jiang 2016). Additionally, optimizing data transfer between the CPU and GPU will be a key focus to minimize delays and ensure efficient utilization of computational resources.

2.1.2 Emphasizing Numerical Accuracy:

Ensuring the accuracy of electromagnetic simulations is paramount to the reliability of our research outcomes. We will prioritize numerical accuracy by validating our simulations against known analytical solutions. Special attention will be given to validating the effectiveness of Uniaxial Perfectly Matched Layer (UPML) boundary conditions (Meng, Wong, S, Macon, C & Jin, J-M 2014). This validation process is crucial to ensuring the reliability and trustworthiness of our simulation results.

2.1.3 Conducting Performance Evaluations:

Once the computational system is configured, we will conduct comprehensive performance evaluations to assess its scalability, efficiency, and computational throughput. Our evaluations will include testing the system's ability to handle large and complex tasks without performance degradation (Demir, V 2014). We will compare the performance of our optimized system with previous methods to evaluate the improvements in accuracy and speed achieved through our research efforts. These performance evaluations will provide valuable insights into the effectiveness of our optimization strategies and guide future research directions.

2.2 Issues and challenges

Identifying and addressing the challenges inherent in collaborative CPU-GPU electromagnetic simulations using the Finite-Difference Time-Domain (FDTD) algorithm is paramount for the project's success.

2.2.1 Balancing Workload Distribution between CPU and GPU:

One significant challenge revolves around balancing the distribution of computational tasks between the CPU and GPU while minimizing data transfer overhead. This necessitates ensuring that both processors contribute their fair share of work without causing delays. To tackle this, implementing dynamic load balancing algorithms can adaptively allocate tasks based on system performance, optimizing resource utilization and computational efficiency (Demir, V 2014). Moreover, optimizing data transfer protocols can mitigate latency issues, facilitating seamless communication between the CPU and GPU.

2.2.2 Algorithm Optimization for GPU Parallel Processing:

Another challenge lies in optimizing algorithms to fully harness the parallel processing power of GPUs while ensuring synchronization and data integrity (Demir, V 2014). Inefficient algorithm design can limit the potential performance gains offered by GPU parallelism. To address this, conducting thorough algorithm profiling and optimization is crucial. Leveraging GPU-specific programming techniques such as CUDA or OpenMp and implementing synchronization mechanisms can ensure data consistency and integrity across parallel threads, thereby enhancing computational efficiency.

2.2.3 Managing GPU Memory Limitations and Optimization:

Efficient management of GPU memory limitations and optimization is also critical. Inadequate memory management can lead to memory allocation errors, performance degradation, and limited scalability. Employing memory optimization techniques such as memory pooling, data compression, and memory reuse can mitigate GPU memory constraints (Morunov, ND & Golovashkin, DL 2019). Furthermore, implementing efficient data structures and algorithms optimized for GPU architecture can improve computational efficiency, ensuring optimal utilization of available resources.

3.2 Methodology:

The methodology involves a systematic approach consisting of four main phases: Setup and Preparation, GPU Implementation, Performance Evaluation, and Documentation and Reporting. During the Setup and Preparation phase, the necessary hardware and software resources will be procured and configured. The GPU Implementation phase will involve adapting the FDTD algorithm for GPU parallelization, optimizing memory usage, and implementing load balancing techniques. Subsequently, the Performance Evaluation phase will include designing experiments, executing simulations, and analyzing results against analytical solutions. Finally, the Documentation and Reporting phase will involve compiling project documentation, preparing technical reports, and presenting findings to the industry and academic supervisors.

3.3 Challenges and strategies

In collaborative CPU-GPU acceleration for electromagnetic simulations using the FDTD algorithm, several challenges emerge, each requiring strategic approaches for resolution. Firstly, managing workload distribution between CPUs and GPUs presents a significant hurdle. To tackle this, a comprehensive understanding of CPU and GPU architectures, synchronization mechanisms, and load balancing strategies is crucial (Ying Xu & Rongling Jiang 2016). Secondly, optimizing GPU algorithms for efficient performance demands dedicated efforts in studying OpenMP programming, experimenting with various parallelization schemes, and leveraging available online resources (Demir, V 2014). Thirdly, ensuring efficient GPU memory management is essential to maximize memory utilization while minimizing wastage. This involves exploring techniques like memory pooling, data compression, and regular monitoring of memory usage patterns ((Ying Xu & Rongling Jiang 2016).

3.5 Contingency planning

The contingency plan for the project involves a structured approach to address challenges encountered.

3.5.1. Identification of Challenges:

The first step is to identify specific challenges encountered during the implementation of the FDTD algorithm. This may include technical limitations, resource constraints, or unexpected obstacles that hinder progress.

3.5.2 Evaluation of Alternatives and Feasibility Assessment:

If difficulties persist and significantly impede the implementation of the FDTD algorithm, we will explore alternative solutions. One such alternative is the adoption of 2D wave models, which are simpler to implement and require fewer computational resources. The industry supervisor and I will assess the feasibility and suitability of transitioning to 2D wave models. This assessment considers factors such as computational efficiency, accuracy of results, and alignment with project objectives.

3.5.3 Consultation:

Throughout the contingency planning process, regular consultations with both the supervisors are scheduled via MS teams. These consultations aim to gather insights and guidance on alternative approaches, including the adoption of 2D wave models.

3.5.4 Transition Planning and Implementation of Alternative Approach:

If the decision is made to pivot to 2D wave models, I will develop a detailed transition plan. This plan outlines the steps required to transition from the FDTD algorithm to 2D wave models, including software adjustments, data migration, and testing procedures. Following the transition plan, the chosen alternative approach, i.e., 2D wave models will be implemented. This involves updating software, adapting simulation methodologies, and ensuring compatibility with existing project infrastructure.

3.5.5 Documentation and Reporting:

Finally, I will document all aspects of the contingency planning process, including challenges encountered, decisions made, and actions taken. This documentation serves as a reference for future projects and facilitates knowledge sharing within the team.

**Maxwell's Equations and Yee's Algorithm**

#### Maxwell's Equations

Maxwell's Equations are a set of coupled vectorial partial differential equations that describe the behavior of electromagnetic fields. They form the foundation of electromagnetics and optics. The equations are as follows:

Where:

Here, \( \epsilon\_r \) and \( \mu\_r \) are the relative permittivity and permeability, respectively; \( \sigma \) and \( \sigma^\* \) are material parameters; \( \mathbf{J}\_{\text{source}} \) and \( \mathbf{M}\_{\text{source}} \) are source terms; and \( \epsilon\_0 \) and \( \mu\_0 \) are constants representing the permittivity and permeability of free space. The variables \( \mathbf{E} \) and \( \mathbf{B} \) represent the electric and magnetic fields, respectively.

Maxwell's Equations contain both space and time derivatives, with space derivatives represented by the curl (\(\nabla \times\)) and divergence (\(\nabla \cdot\)) operators.

#### Yee's Algorithm in FDTD

The Finite-Difference Time-Domain (FDTD) method is a direct numerical integration technique that transforms the partial derivatives in Maxwell's Equations into finite differences on a discrete grid. This grid contains the material parameters and represents the problem to be solved. The resulting equations are integrated numerically in both space and time using initial and boundary conditions.

Yee's algorithm, introduced in 1966, provides a second-order accurate FDTD solution for Maxwell's Equations. It uses an intertwined grid where the electric field components are surrounded by four magnetic field components and vice versa. This algorithm ensures accurate representation of the fields and their interactions.

The space and time derivatives are approximated as follows:

- The first spatial derivative of a field \( u \) is:

\[

\frac{\partial u}{\partial x} \bigg|\_{(i \Delta x, j \Delta y, k \Delta z, n \Delta t)} \approx \frac{u^n\_{i+1/2,j,k} - u^n\_{i-1/2,j,k}}{\Delta x}

\]

- The first time derivative is:

\[

\frac{\partial u}{\partial t} \bigg|\_{(i \Delta x, j \Delta y, k \Delta z, n \Delta t)} \approx \frac{u^{n+1/2}\_{i,j,k} - u^{n-1/2}\_{i,j,k}}{\Delta t}

\]

These transformations are applied to the first two Maxwell's Equations. The other two equations are inherently satisfied by the algorithm's structure. By using these difference formulas and simple averaging, we derive six explicit time-stepping equations. The electric and magnetic fields are updated using these equations.

#### Initial and Boundary Conditions

FDTD simulations require initial conditions for the field values and boundary conditions at the spatial boundaries. Typically, all field values are initialized to zero. It's also possible to save the field values between time steps and resume the calculation from the saved state.

To prevent unwanted reflections at the boundaries of the simulation domain, Absorbing Boundary Conditions (ABCs) are used. In this setup, Liao and Uniaxial Perfectly Matched Layer (UPML) boundary conditions are employed. Liao's ABC uses a Newton extrapolation polynomial technique to simulate "endless free space," absorbing electromagnetic energy and eliminating reflections. The UPML ABC is an advanced version of Berenger's Perfectly Matched Layer (PML) technique, which involves an artificial material slab designed to absorb incoming electromagnetic energy effectively. The absorbance of the material increases polynomially deeper into the PML absorber slab, ensuring efficient energy absorption.

4 EXPERIMENTAL SETUPS

The experimental setup for this study involves a 1D FDTD simulation to model the propagation of an electromagnetic wave. The simulation is configured with the following parameters:

* Grid Size (SIZE): 200 cells
* Maximum Time Steps (MAX\_TIME): 250
* Courant Number (COURANT\_NUMBER): 0.5
* Spatial Step (DELTA\_X): 0.01 meters
* Time Step (DELTA\_T): Calculated based on the Courant number and the speed of light

The initial conditions include setting the electric field and magnetic field arrays to zero. A Gaussian pulse is introduced at the first grid point (ez[0]) to act as the source of the electromagnetic wave.

The simulation is integrated with Gnuplot for real-time visualization of the electric field distribution. This allows for observing the wave dynamics as the simulation progresses.

2D FDTD Simulation

This section aims to simulate the propagation of electromagnetic waves within a 2D domain using the FDTD method. The setup involves initializing various field parameters, updating the electromagnetic fields over time, and visualizing the results through appropriate software tools.

Simulation Domain and Parameters

The simulation domain is structured as a 200x200 cell grid, representing a 2D spatial area. The simulation runs for a total of 350-time steps to observe the wave propagation adequately. The source of the electromagnetic waves is placed at the center of the grid, at coordinates (100, 100). The type of wave generated by the source can be configured to be either Gaussian, sine, or impulse, controlled by preprocessor directives (GAUSSIAN, SINE, IMPULSE).

Constants

Several physical constants are used in the simulation. The speed of light, \(C\), is set to \(3 \times 10^8\) m/s. The permittivity of free space, \(\epsilon\_0\), is defined as \(\frac{1}{36\pi} \times 10^{-9}\) F/m, and the permeability of free space, \(\mu\_0\), is \(4\pi \times 10^{-7}\) H/m. The grid spacing, \(\Delta\), is set to \(1 \mu m\), ensuring a fine resolution for the simulation. For sine wave sources, a frequency of \(1.5 \times 10^{13}\) Hz is used.

#### Initialization

Initially, the electric field components (\(E\_z, E\_{zx}, E\_{zy}\)) and magnetic field components (\(H\_x, H\_y\)) are set to zero across the entire domain. The permittivity and permeability are uniformly set to \(\epsilon\_0\) and \(\mu\_0\) respectively. Conductivity matrices (\(\sigma\_x, \sigma\_y, \sigma^\*\_x, \sigma^\*\_y\)) are also initialized to zero, playing a crucial role in the boundary condition implementation.

#### Boundary Conditions

To minimize reflections back into the domain and simulate an infinite space, Perfectly Matched Layer (PML) boundary conditions are applied. These conditions absorb outgoing waves effectively. The PML is 25 cells wide with a grading order of 6. The maximum conductivity \(\sigma\_{\text{max}}\) is computed to ensure a reflection coefficient of \(10^{-6}\), providing an efficient absorption mechanism at the boundaries.

#### Source Initialization

Depending on the configuration, the source generates either a Gaussian pulse, a sine wave, or an impulse. For a sine wave, the source at the center of the grid generates waves at the specified frequency. For a Gaussian pulse, the waveform is calculated based on the simulation parameters. If an impulse is configured, a unit-time step impulse is generated, initiating the wave propagation.

#### Field Updates

The fields are updated at each time step using the FDTD method. Initially, the magnetic fields (\(H\_x, H\_y\)) are updated, followed by the electric fields (\(E\_z, E\_{zx}, E\_{zy}\)). This updating process is parallelized using OpenMP to enhance computational efficiency, allowing for faster simulations on multi-core processors.

#### Data Collection

During each time step, the electric field \(E\_z\) data is recorded and saved to files. This data is used for subsequent analysis and visualization. Additionally, the time taken for each update step is recorded and written to a file, providing insights into the computational performance of the simulation.

#### Visualization

GNUPlot is employed for visualizing the simulation results. The electric field data at each time step is plotted, illustrating the propagation of the electromagnetic waves through the 2D domain. This visualization helps in understanding the wave dynamics and interactions within the simulated environment.

#### Conclusion

This experimental setup facilitates a detailed study of electromagnetic wave propagation in a 2D domain, offering valuable insights into wave behavior under various boundary conditions and source configurations. The use of the FDTD method and PML boundary conditions ensures accurate simulation results, making this setup a robust tool for research in electromagnetics.

5 RESULTS

The simulation models the propagation of electromagnetic waves in a one-dimensional space. The real-time visualization using Gnuplot allows for immediate observation of the wave dynamics. Initially, a Gaussian pulse is introduced at the source node, which propagates through the medium as time progresses. The electric field (Ez) and magnetic field (Hy) interact according to Maxwell’s equations, showing the characteristic behavior of wave propagation, reflection, and potential interference.

|  |
| --- |
|  |
| Figure? : 1D Animation of FDTD algorithm in space |

Throughout the simulation, the electric field values at each spatial point are plotted, providing a clear visualization of the wave's movement and changes over time. This visual representation confirms that the FDTD method accurately captures the dynamics of electromagnetic wave propagation. The results demonstrate the effectiveness of the FDTD algorithm in modeling wave behavior, and the integration with Gnuplot proves useful for real-time analysis and visualization.

The main limitation of the FDTD method is the restriction imposed by the CFL stability condition, which necessitates small time steps for stability. This constraint can be addressed by advanced methods such as the Crank-Nicolson FDTD (CN-FDTD) or the Alternating Direction Implicit FDTD (ADI-FDTD), which provide unconditional stability but require significantly more computational resources.

|  |
| --- |
|  |

|  |
| --- |
|  |
| Performance evaluation  To assess the practical benefits of the GPU-based Finite-Difference Time-Domain (FDTD) implementation compared to the CPU implementation, we will evaluate the performance based on the total execution time for a specified domain size and number of time steps, as well as the number of cell operations completed per second. |
| |  | | --- | |  | |

|  |
| --- |
|  |

|  |
| --- |
|  |

6 DISCUSSIONS

Usability

While this version of the FDTD method implementation may not be ready for real-time applications since various constraints from real world were ignored but still adopting a GPU-based approach results in a significant performance enhancement. This improvement not only increases productivity but also it will assist in facilitating the development of higher-quality products. Furthermore, by eliminating the limitations and constraints associated with script-only modeling approaches found in other software packages, this implementation offers improved usability. This enhances the accessibility of the application to a wider audience, including students with varying levels of programming expertise.

Future work

6 CONCLUSIONS

REFERENCES

1. Ying Xu, Huimin Ma & Rongling Jiang 2016, ‘Collaborating CPU and GPU for the electromagnetic simulations with the FDTD algorithm: COLLABORATING CPU AND GPU FOR FDTD ALGORITHM’, *Concurrency and Computation*, vol. 29, pp. e3859-.
2. Meng, H-T, Nie, B-L, Wong, S, Macon, C & Jin, J-M 2014, GPU accelerated finite-element computation for electromagnetic analysis, IEEE antennas & propagation magazine, vol. 56, IEEE, PISCATAWAY, no. 2, pp. 39–62.
3. Chi, J, Liu, F, Weber, E, Li, Y & Crozier, S 2011, ‘GPU-Accelerated FDTD Modeling of Radio-Frequency Field-Tissue Interactions in High-Field MRI’, *IEEE Transactions on Biomedical Engineering*, vol. 58, no. 6, pp. 1789–1796.
4. Demir, V 2014, ‘A simple GPU implementation of FDTD/PBC algorithm for analysis of periodic structures’, *Applied Computational Electromagnetics Society Journal*, vol. 29, no. 12, pp. 1018–1024.
5. Morunov, ND & Golovashkin, DL 2019, ‘Design features of block algorithms of FDTD-method implemented on a GPU using MATLAB’, *Kompʹûternaâ Optika*, vol. 43, no. 4, pp. 671–676.
6. John B. Schneider 2010, *Understanding the FDTD Method*. [online] Available at: https://eecs.wsu.edu/~schneidj/ufdtd/.
7. ‌Lively, S. David, "GoLightly : A GPU Implementation of the Finite-Difference Time-Domain Method" (2022). Electrical Engineering Theses and Dissertations. 54. https://scholar.smu.edu/engineering\_electrical\_etds/54

Appendix