

Optimizing an FPGA-based 3D FDTD Accelerator through HLS

Tan Jin Hung

January 2, 2026

Abstract

Contents

| | | |
|----------|---|-----------|
| 1 | Introduction | 2 |
| 2 | Background | 3 |
| 2.1 | Maxwell's Equation | 3 |
| 2.2 | The FDTD Method | 4 |
| 2.2.1 | Yee's Staggered Grid | 5 |
| 2.2.2 | Temporal Update Equations | 6 |
| 2.3 | Field-Programmable Gate Array (FPGA) Architecture | 7 |
| 2.4 | High-Level Synthesis (HLS) | 9 |
| 3 | Methodology | 10 |
| 3.1 | Baseline Implementation | 10 |
| 3.2 | Domain Decomposition | 10 |
| 3.3 | Memory Subsystem Optimization | 10 |
| 3.4 | Advanced Throughput Optimization | 10 |
| 4 | Evaluation | 10 |
| 4.1 | Environments | 10 |
| 4.2 | Results | 10 |
| 5 | Discussion | 10 |
| 6 | Conclusion | 10 |

1 Introduction

The current technological era has enabled the use of computers to simulate our complex understanding of the physical world. However, simulations that accurately mimic real-life phenomena remain computationally intensive for many practical applications. To manage this complexity, these simulations are often decomposed down into their fundamental physical components. Among these fundamentals, the simulation of electromagnetic waves remains at the forefront of modern research, with the FDTD method serving as a primary tool for high-fidelity analysis in complex environments [1].

A primary method for simulating electromagnetic waves is the Finite-Difference Time-Domain (FDTD) method [2]. While FDTD is a cornerstone of computational electromagnetics, providing a robust foundation for analysis, it faces significant challenges in 3D space. Specifically, the iterative nature of the method creates bottlenecks in memory bandwidth and data throughput, often characterized as the ‘memory wall’ in high-performance computing [3, 4]. As noted by Kong and Su [5], the computational demand of the 3D FDTD algorithm increases cubically with the grid resolution, necessitating highly parallel hardware architectures to maintain feasible simulation times. Traditionally, these simulations have been offloaded to Graphics Processing Units (GPUs). However, Field-Programmable Gate Arrays (FPGAs) have emerged as a compelling alternative. FPGAs offer distinct advantages over GPUs, including deterministic latency, superior energy efficiency [3], and the ability to implement highly customizable memory architectures for parallelism. Recent research has shown that through customized memory architectures, FPGAs can outperform GPUs in energy-per-update for structured-mesh solvers [6].

Despite these advantages, FPGAs are traditionally difficult to program using Hardware Description Language (HDLs). Implementing an FDTD simulation in HDL is tedious and requires the manual design of standard components that are better suited for automation. High-Level Synthesis (HLS) addresses these challenges by allowing designers to describe complex hardware architectures using high-level languages such as C/C++. Recent benchmarking has demonstrated that HLS can achieve performance parity with manual RTL design while significantly reducing the development cycle [7, 8].

This paper presents an optimized 3D FDTD accelerator designed via HLS, exploring several architectural optimization strategies, including:

Spatial Tiling Maximizing memory bandwidth by utilizing on-chip Block Random Access Memories (BRAMs) to store local stencil and reduce

external memory access.

HLS Directives Leveraging compiler pragmas to optimize loop pipelining and unrolling, thereby improving the throughput of the kernel.

Task-Level Parallelism Decomposing the algorithm into concurrent tasks to increase parallelism.

Through these optimization, we demonstrate that an HLS-driven approach can also produce a high-performance FDTD accelerator, while significantly improving code readability, maintainability, and design iteration speed compared to traditional HDL-based workflows.

2 Background

This section establishes the theoretical and technical foundation required to design and implement an FPGA-based 3D FDTD accelerator. To provide a complete context for the methodology, the discussion begins with the governing physics of electromagnetic wave propagation through Maxwell's Equations. Subsequently, it details the transition from continuous calculus to a discrete numerical model using Yee's staggered grid and the resulting leapfrog update equations. Finally, the section explores the architectural characteristics of Field-Programmable Gate Arrays (FPGAs) and the High-Level Synthesis (HLS) design flow. This hardware background is essential for understanding how algorithmic bottlenecks are addressed through spatial tiling and parallel execution, which are the primary focus of this work.

2.1 Maxwell's Equation

The underlying physics governing electromagnetic waves are described by the interaction of electric and magnetic fields, a theory unified by J. C. Maxwell [9]. Maxwell consolidated the independent laws of Faraday, Ampère, and Gauss, demonstrating that electric and magnetic fields are interdependent. While the original theory consisted of 20 equations, it was later refined by O. Heaviside into the modern four-vector representation. These equations, representing Gauss's Law, Gauss's Law for Magnetism, Faraday's Law, and Ampère's Law, are given by:

$$\nabla \cdot \mathbf{D} = \rho_v \quad (1a)$$

$$\nabla \cdot \mathbf{B} = 0 \quad (1b)$$

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \quad (1c)$$

$$\nabla \times \mathbf{H} = \frac{\partial \mathbf{D}}{\partial t} + \mathbf{J} \quad (1d)$$

where the constitutive relations for linear, isotropic media are:

$$\mathbf{D} = \epsilon \mathbf{E} \quad (1e)$$

$$\mathbf{B} = \mu \mathbf{H} \quad (1f)$$

$$\mathbf{J} = \sigma \mathbf{E} \quad (1g)$$

While (1) provides a complete description of electromagnetism, it is more practical for numerical wave propagation analysis to substitute the constitutive relations directly into the curl equations. By substituting the flux densities (\mathbf{D}, \mathbf{B}) and the current density (\mathbf{J}) into the curl equations, we obtain the coupled system of first-order partial differential equations:

$$\nabla \times \mathbf{E} = -\mu \frac{\partial \mathbf{H}}{\partial t} - \sigma_m \mathbf{H} \quad (2a)$$

$$\nabla \times \mathbf{H} = \epsilon \frac{\partial \mathbf{E}}{\partial t} + \sigma \mathbf{E} \quad (2b)$$

This representation shows explicitly the interaction between the electric (\mathbf{E}) and magnetic (\mathbf{H}) field intensities, providing the fundamental framework for the FDTD method's time-stepping procedure. By rearranging (2) to isolate the temporal derivatives, we obtain the form used for numerical integration:

$$\frac{\partial \mathbf{H}}{\partial t} = -\frac{1}{\mu} (\nabla \times \mathbf{E} + \sigma_m \mathbf{H}) \quad (3a)$$

$$\frac{\partial \mathbf{E}}{\partial t} = \frac{1}{\epsilon} (\nabla \times \mathbf{H} - \sigma \mathbf{E}) \quad (3b)$$

2.2 The FDTD Method

To solve the coupled system of equations in (3) using a digital computer, the continuous temporal and spatial derivatives must be transformed into a discrete form. The Finite-Difference Time-Domain (FDTD) method, first

proposed by Kane S. Yee in 1966 [2], achieves this by employing central-difference approximations. This approach maps the electric and magnetic fields onto a discrete staggered grid, allowing the system to be solved iteratively through a "leapfrog" time-stepping procedure. To ensure the numerical stability of this iterative scheme, the time step must satisfy the Courant-Friedrichs-Lowy (CFL) condition [10].

2.2.1 Yee's Staggered Grid

To numerically solve Maxwell's equations, the continuous spatial and temporal domains must be discretized. The FDTD method utilizes the Yee staggered grid, an arrangement where electric (**E**) and magnetic (**H**) field components are spatially interleaved within a unit cell, as shown in Figure 1, where the **E** field and **H** field are staggered by half a temporal step.

In this configuration, each **E** component is located at the center of the edges of a grid cell, while each **H** component is positioned at the center of the faces. The spatial coordinates of the field components within a single Yee cell are summarized in Table 2.2.1.

Table 1: Spatial offsets for **E** and **H** field components.

| Axis | Electric Field (E) | Magnetic Field (H) |
|------|---------------------------|---|
| x | $(i + \frac{1}{2}, j, k)$ | $(i, j + \frac{1}{2}, k + \frac{1}{2})$ |
| y | $(i, j + \frac{1}{2}, k)$ | $(i + \frac{1}{2}, j, k + \frac{1}{2})$ |
| z | $(i, j, k + \frac{1}{2})$ | $(i + \frac{1}{2}, j + \frac{1}{2}, k)$ |

This spatial staggering ensures that each field component is surrounded by four circulating dual-field components. This layout provides a natural representation of the curl operators, enabling second-order accurate central-difference approximations. From a hardware perspective, this arrangement creates a 3D stencil dependency; updating a single point requires values from its neighbors, which necessitates efficient on-chip memory management to maintain high throughput.

While the Yee grid defines the spatial distribution of the fields, the FDTD method also requires discretization in the temporal domain. To achieve a stable and accurate simulation, a "leapfrog" time-stepping scheme is employed. As illustrated in Figure 1, the magnetic fields are offset by half a temporal step relative to the electric fields, allowing the interleaved calculation of field updates.

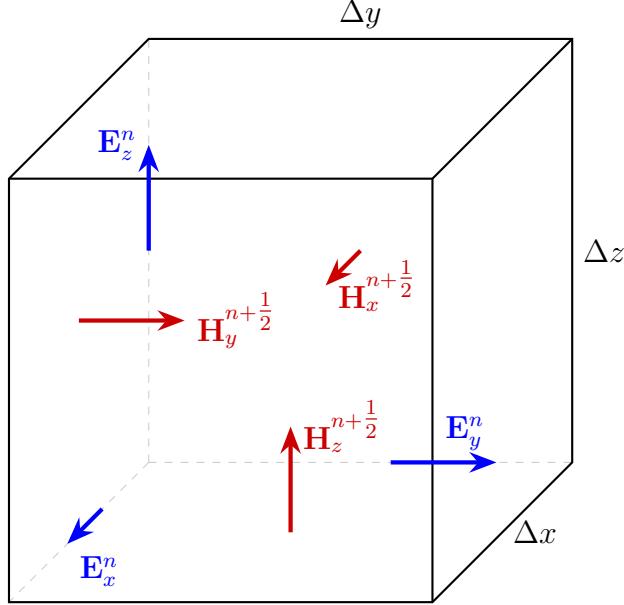


Figure 1: Spatial arrangement of electric (\mathbf{E}) and magnetic (\mathbf{H}) field components. Dimensions $\Delta x, \Delta y, \Delta z$ represent grid increments, and n denotes the temporal step.

2.2.2 Temporal Update Equations

In addition to spatial staggering, the FDTD method employs temporal interleaving, commonly referred to as the "leapfrog" scheme as described by Taflove [1]. Under this arrangement, the \mathbf{E} components are updated at integer time steps ($n, n + 1, \dots$), whereas the \mathbf{H} components are updated at half-integer time steps ($n + 1/2, n + 3/2, \dots$).

By applying central-difference approximations to the temporal and spatial derivatives in Maxwell's Equations, we obtain the explicit update equations. For a general lossy medium, the update for a single component such as H_x is expressed in Equation 4:

$$\begin{aligned} \mathbf{H}_x^{n+1/2} [i, j, k] &= \left(\frac{1 - \frac{\sigma_m \Delta t}{2\mu}}{1 + \frac{\sigma_m \Delta t}{2\mu}} \right) \mathbf{H}_x^{n-1/2} [i, j, k] \\ &\quad + \left(\frac{\frac{\Delta t}{\mu}}{1 + \frac{\sigma_m \Delta t}{2\mu}} \right) \left[\frac{\mathbf{E}_y^n|_{k+1} - \mathbf{E}_y^n|_k}{\Delta z} - \frac{\mathbf{E}_z^n|_{j+1} - \mathbf{E}_z^n|_j}{\Delta y} \right] \end{aligned} \quad (4)$$

In the case of a lossless, homogeneous medium such as free space, the conductivities σ and σ_m are zero. This reduces the decay coefficients to unity,

and the update equations for the full 3D system can be simplified to:

$$\mathbf{H}_x^{n+1/2} = \mathbf{H}_x^{n-1/2} + \frac{\Delta t}{\mu_0} \left[\frac{\delta \mathbf{E}_y^n}{\Delta z} - \frac{\delta \mathbf{E}_z^n}{\Delta y} \right] \quad (5a)$$

$$\mathbf{H}_y^{n+1/2} = \mathbf{H}_y^{n-1/2} + \frac{\Delta t}{\mu_0} \left[\frac{\delta \mathbf{E}_z^n}{\Delta x} - \frac{\delta \mathbf{E}_x^n}{\Delta z} \right] \quad (5b)$$

$$\mathbf{H}_z^{n+1/2} = \mathbf{H}_z^{n-1/2} + \frac{\Delta t}{\mu_0} \left[\frac{\delta \mathbf{E}_x^n}{\Delta y} - \frac{\delta \mathbf{E}_y^n}{\Delta x} \right] \quad (5c)$$

$$\mathbf{E}_x^{n+1} = \mathbf{E}_x^n + \frac{\Delta t}{\varepsilon_0} \left[\frac{\delta \mathbf{H}_z^{n+1/2}}{\Delta y} - \frac{\delta \mathbf{H}_y^{n+1/2}}{\Delta z} \right] \quad (5d)$$

$$\mathbf{E}_y^{n+1} = \mathbf{E}_y^n + \frac{\Delta t}{\varepsilon_0} \left[\frac{\delta \mathbf{H}_x^{n+1/2}}{\Delta z} - \frac{\delta \mathbf{H}_z^{n+1/2}}{\Delta x} \right] \quad (5e)$$

$$\mathbf{E}_z^{n+1} = \mathbf{E}_z^n + \frac{\Delta t}{\varepsilon_0} \left[\frac{\delta \mathbf{H}_y^{n+1/2}}{\Delta x} - \frac{\delta \mathbf{H}_x^{n+1/2}}{\Delta y} \right] \quad (5f)$$

where δ denotes the central-difference operator (e.g., $\delta \mathbf{E}_y^n = \mathbf{E}_y^n|_{k+1} - \mathbf{E}_y^n|_k$, where the index +1 refers to the spatial increment along the corresponding axis).

This representation reveals that the field update is a function of the previous state and the surrounding spatial curl, scaled by a material-dependent ratio ($\Delta t/\mu_0$ or $\Delta t/\varepsilon_0$). These ratios, combined with the grid spacing Δs , form the basis for the normalized coefficients used in hardware acceleration.

2.3 Field-Programmable Gate Array (FPGA) Architecture

While the 3D FDTD algorithm possesses significant inherent symmetry and parallelism, its performance is highly dependent on the ability of the underlying hardware to exploit these features. Traditionally, Graphics Processing Units (GPUs) have dominated this field due to their massive SIMD (Single Instruction, Multiple Data) parallelism. However, FPGAs have emerged as a powerful alternative for accelerating computational electromagnetics.

The primary advantage of an FPGA-based design over a GPU lies in its architectural flexibility. Unlike the fixed memory hierarchy of a GPU, an FPGA allows the designer to implement a fully customized memory architecture, as noted in [6]. This capability is essential for 3D FDTD, as it enables the creation of application-specific data paths and local memory buffers that can exploit the spatial dependencies of the Yee stencil. By tailoring

the hardware to the algorithm, FPGAs can achieve high throughput with lower power consumption and deterministic latency. As noted by Zohouri et al. [4], the advantage of FPGAs in stencil-based high-performance computing lies in their ability to exploit fine-grained parallelism to achieve high performance-per-clock ratios.

To implement the complex update equations defined in Equation (5), the FPGA fabric utilizes several specialized hardware primitives:

- **Configurable Logic Blocks (CLBs):** Consisting of Look-Up Tables (LUTs) and Flip-Flops (FF), these are the fundamental building blocks of the FPGA fabric. They implement the "glue logic" and control flow of the system. In an HLS-driven design, LUTs are used to build the Finite State Machines (FSMs) that coordinate dataflow between BRAM buffers and DSP slices. An increase in available LUTs allows for greater control complexity and more sophisticated optimization strategies.
- **Digital Signal Processing (DSP) Slices:** These are dedicated silicon blocks optimized for high-speed arithmetic. Each slice typically consists of a pre-defined set of operations, allowing field updates to be performed in a single clock cycle without consuming vast amounts of general-purpose logic. An increase in DSP count directly translates to higher throughput, as more field components can be updated in parallel. Architecture studies in [5] demonstrate that the spatial parallelism of FPGAs is uniquely suited to the concurrent field updates of the 3D FDTD grid.
- **Block RAM (BRAM):** These represent the on-chip memory resources that distinguish FPGAs from fixed-cache architectures. Unlike standard global memory (DRAM), BRAM can be "partitioned" and "reshaped" to provide multiple independent access ports. This is a critical feature for stencil computations, where the hardware must fetch several neighboring values simultaneously. Increasing BRAM capacity allows larger grid volumes to be stored on-chip, reducing the need for high-latency external memory access. Recent methodologies have focused on automatically extracting these memory patterns to reduce the overall FPGA memory footprint for complex stencil codes [11].

The performance and scalability of an FPGA-based FDTD accelerator are directly constrained by the availability of these resources. Since the 3D FDTD algorithm is both computationally and memory-intensive, there is a fundamental trade-off between the simulation volume and the update speed. A higher count of DSP slices allows for more Processing Elements (PEs)

to operate in parallel, while an abundance of BRAM enables the storage of larger grid dimensions on-chip, thereby avoiding the significant latency penalties associated with external DDR memory access. Consequently, an optimized design must balance the utilization of these resources to maximize the throughput, measured in Millions of cells per second (Mcells/s), while remaining within the physical limits and memory bandwidth constraints of the target device.

2.4 High-Level Synthesis (HLS)

While FPGA architectures provide the necessary primitives for acceleration, manual implementation of large-scale projects in low-level Hardware Description Languages (HDLs) is labor-intensive and error-prone. Many hardware structures, such as memory controllers and nested loop pipelines, follow standard patterns that are better suited for automation than manual coding. High-Level Synthesis (HLS) addresses these challenges by acting as an abstraction layer that automates the generation of these standard structures [12]. Furthermore, HLS allows designers to describe complex projects using high-level languages like C or C++, significantly lowering the barrier of entry compared to RTL design in Verilog or VHDL.

To bridge the gap between algorithmic description and hardware realization, HLS provides several critical optimization directives and data-handling paradigm [12]:

- **Pipelining (#pragma HLS PIPELINE):** This is a fundamental optimization for FDTD. It enables the concurrent execution of different stages of the update equation. By pipelining the inner loops, the hardware can begin a new field update before the previous one has completed, with the primary objective of achieving an Initiation Interval (II) of 1.
- **Memory Partitioning (#pragma HLS ARRAY_PARTITION):** Standard BRAM is limited by its physical port count (typically two). Since 3D FDTD stencils require multiple simultaneous operands, HLS allows for the "partitioning" of these arrays into smaller, independent memory banks. This provides the parallel access ports necessary to feed the DSP update engines without stalling the pipeline, a technique widely utilized to overcome the memory bottlenecks of structured-mesh solvers [6, 11].
- **Task-Level Parallelism (#pragma HLS DATAFLOW):** This directive enables the execution of functions in a producer-consumer pattern. In the context of this work, it allows the electric and magnetic field

update functions to operate concurrently, overlapping their execution to maximize the utilization of the FPGA’s spatial resources.

- **Streaming Interfaces (`hls::stream` or `hls::stream_of_blocks`):** Beyond standard memory-mapped access, HLS supports a streaming data paradigm through built-in libraries (`hls_streamofblocks.h` or `hls_stream.h`). Using FIFO (First-In-First-Out) buffers, data can be passed point-to-point between hardware modules without the overhead of global memory addressing. In 3D FDTD, streaming is essential for moving planes of data through the update engine, minimizing latency and enabling the high-throughput ”Stream of Blocks” architecture[13].

By utilizing these directives, HLS enables a ”stepwise refinement” design process. A designer can begin with a baseline software implementation and iteratively apply hardware constraints to transform a sequential C-program into a high-performance, parallelized 3D FDTD accelerator. As shown in automation frameworks such as SASA [14], overlapping computation and communication optimizations are critical for scaling FDTD accelerators to handle massive data volume of 3D electromagnetic simulations.

3 Methodology

- 3.1 Baseline Implementation
- 3.2 Domain Decomposition
- 3.3 Memory Subsystem Optimization
- 3.4 Advanced Throughput Optimization

4 Evaluation

- 4.1 Environments
- 4.2 Results

5 Discussion

6 Conclusion

References

- [1] A. Taflove and S. C. Hagness, *Computational Electromagnetics: The Finite-Difference Time-Domain Method*, 3rd ed. Boston, MA: Artech House, 2005, the definitive reference for FDTD theory and implementation.
- [2] K. S. Yee, “Numerical solution of initial boundary value problems involving maxwell’s equations in isotropic media,” *IEEE Transactions on Antennas and Propagation*, vol. 14, no. 3, pp. 302–307, 1966.
- [3] T. Nguyen *et al.*, “FPGA-based HPC accelerators: An evaluation on performance and energy efficiency,” *Concurrency and Computation: Practice and Experience*, vol. 34, no. 18, 2022.
- [4] H. R. Zohouri *et al.*, “High-performance computing with FPGAs: Case study in optimizing stencil kernels,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 5, pp. 975–989, 2018.
- [5] C. Kong and T. Su, “Parallel hardware architecture of the 3d FDTD algorithm with convolutional perfectly matched layer boundary condition,” *IEEE Transactions on Antennas and Propagation*, vol. 64, no. 9, pp. 3936–3946, 2016, provides a comprehensive analysis of the parallel data paths required for 3D FDTD.
- [6] K. Kamalakkannan, “High-level FPGA accelerator design for structured-mesh-based numerical solvers,” Ph.D. dissertation, University of Warwick, 2023. [Online]. Available: <https://wrap.warwick.ac.uk/180089/>
- [7] Y. Bai *et al.*, “Towards a comprehensive benchmark for high-level synthesis targeted to FPGAs,” in *37th Conference on Neural Information Processing Systems (NeurIPS)*, 2023.
- [8] G. Rodriguez-Canal, N. Brown *et al.*, “Stencil-hmls: A multi-layered approach to the automatic optimisation of stencil codes on FPGA,” in *Proceedings of the SC ’23 Workshops*, 2023.
- [9] J. C. Maxwell, *A Treatise on Electricity and Magnetism*, 1st ed. Oxford: Clarendon Press, 1873.
- [10] R. Courant, K. Friedrichs, and H. Lewy, “On the partial differential equations of mathematical physics,” *IBM Journal of Research and Development*, vol. 11, no. 2, pp. 215–234, 1967.

- [11] R. Szafarczyk, S. W. Nabi, and W. Vanderbauwhede, “Reducing FPGA memory footprint of stencil codes through automatic extraction of memory patterns,” in *2022 32nd International Conference on Field-Programmable Logic and Applications (FPL)*, 2022.
- [12] Advanced Micro Devices, Inc., *Vitis HLS User Guide*, v2025.2 ed., AMD, November 2025, uG1399.
- [13] H. K.-H. So *et al.*, “Throughput-oriented FPGA acceleration of FDTD,” in *2018 International Conference on Field-Programmable Technology (FPT)*, 2018.
- [14] X. Tian, Z. Ye, A. Lu, L. Guo, Y. Chi, and Z. Fang, “SASA: A scalable and automatic stencil acceleration framework for optimized hybrid spatial and temporal parallelism on HBM-based FPGAs,” *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 16, no. 2, pp. 1–33, 2023, provides a framework for optimizing 3D stencils by balancing spatial and temporal parallelism.