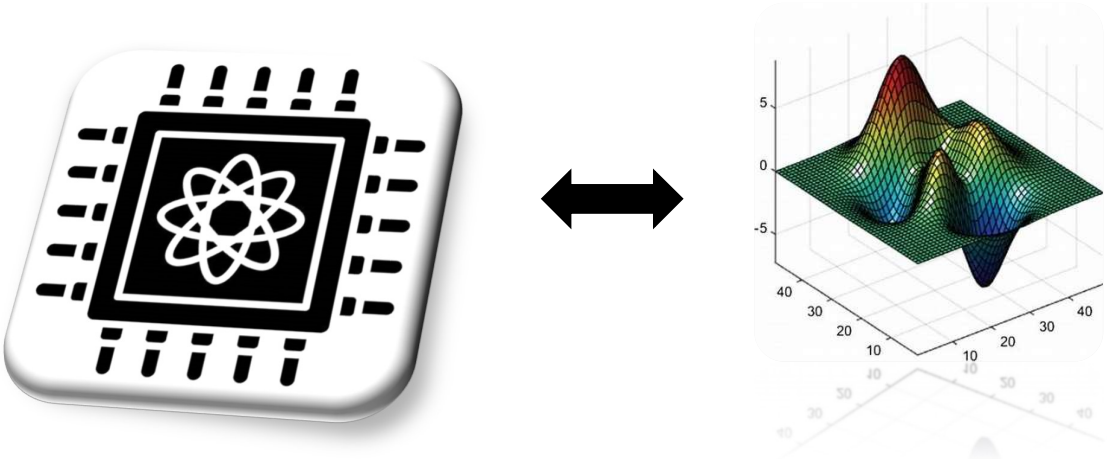
# Centre for Hardware Acceleration & Chip Security

Department of ECE, PES University.

PROJECT REPORT

# HARDWARE ACCELERATORS FOR SOLVING PARTIAL DIFFERENTIAL EQUATIONS



## Team Composition

1. SUDEENDRA KUMAR K (Project Mentor)
2. CHANDAN.M
3. ANIRUDH KULKARNI

CHANDAN.M **PES1UG22EC059**
ANIRUDH KULKARNI **PES1UG22EC035**

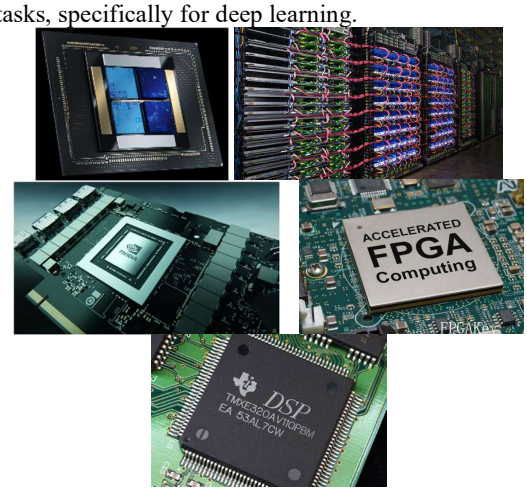
## 1. Abstract-

**This project focuses on the design and implementation of a hardware accelerator architecture, FDMAX, to efficiently solve Partial Differential Equations (PDEs) using the Finite Difference Method (FDM). The architecture comprises a reconfigurable Processing Element (PE) array that adapts to different grid sizes and equations. By optimizing data reuse and minimizing interconnection overhead, FDMAX demonstrates substantial performance improvements and energy savings compared to traditional CPU and GPU implementations.**

## 2. INTRODUCTION

Hardware accelerators are specialized hardware components designed to perform specific computational tasks more efficiently than general-purpose processors (such as CPUs). They are typically used to speed up processing for tasks that require high computational power, such as graphics rendering, machine learning, scientific simulations, data encryption, and signal processing, etc. The efficiency of hardware accelerators comes from their ability to perform specific tasks with optimized architecture, parallelism, and tailored logic. By offloading specific tasks to these accelerators, the overall system can achieve higher performance, lower power consumption, and better utilization of resources. Examples of hardware accelerators include:

- **Graphics Processing Units (GPUs)**: Designed for parallel processing, commonly used in graphics rendering and more recently for machine learning and scientific computations.
- **Field-Programmable Gate Arrays (FPGAs)**: Programmable devices that can be configured to perform specific tasks, offering flexibility and efficiency.
- **Application-Specific Integrated Circuits (ASICs)**: Custom-designed chips for a specific application, providing maximum performance and efficiency but lacking flexibility.
- **Digital Signal Processors (DSPs)**: Specialized for processing digital signals, often used in telecommunications, audio processing, and other signal processing tasks.
- **Tensor Processing Units (TPUs)**: Custom hardware developed by Google for accelerating machine learning tasks, specifically for deep learning.




Complex mathematical computations such as Matrix multiplication, solving Linear Equations, Integrals and Differential equations and Partial Differential Equations (PDEs), etc are fundamental in science and technology. PDEs, in particular, are used to describe various natural and engineering phenomena, including fluid dynamics, heat transfer, and wave propagation. Efficiently solving these equations is crucial for simulations, predictions, and analyses in various fields. Hardware accelerators are increasingly used in scientific and technical fields to accelerate complex calculations. Hardware accelerators can significantly improve the efficiency of mathematical computations by offering:

- **Increased computational speed:** They can execute specialized operations faster than general-purpose CPUs.
- **Energy efficiency:** Optimized for specific tasks, they consume less power.
- **Scalability:** Accelerators can be scaled to handle large data sets and complex problems.

Due to this, Specialized architectures are necessary because general-purpose processors may not efficiently handle the high computational demands and memory bandwidth required for sophisticated tasks. Hardware accelerators can be designed to optimize data paths, memory access patterns, and computation methods, providing a tailored solution for specific problems. Solving PDEs is essential for:

- **Scientific research:** Understanding natural phenomena and processes.
- **Engineering applications:** Designing and optimizing systems in fields like aerospace, mechanical, and civil engineering.
- **Simulations:** Weather forecasting, climate modelling, and more.

In this work, we will be working and implementing FDMAX, an elastic accelerator for partial difference equation with different tools and verify its functionality.

## 3. HOW ARE PDEs SOLVED IN COMPUTERs, AND WHAT CHALLENGES DO CPUS FACE?

PDEs can be solved majorly via two methods; Analytical and Numerical. Numerical method is the one which is most suit for hardware implementation. In Numerical methods, there's numerous ways to solve PDE like;

- **Finite Difference Method (FDM):** Approximates PDEs using discrete differences. It is widely used due to its simplicity and effectiveness.
- **Finite Element Method (FEM) and Finite Volume Method (FVM):** Approximates PDEs using small finite geometries like triangle or tetrahedral.

**Solution Methods**: Solving PDEs can be complex and often requires specialized methods, including:

- **Analytical Solutions:** These are exact solutions expressed in closed form using mathematical functions. Examples: Separation of variables, transform methods (Fourier, Laplace), and method of characteristics. Analytical solutions are usually limited to simple, idealized problems with specific boundary and initial conditions.
- **Numerical Solutions:** These approximate the solution of PDEs using discretization methods. Numerical solutions are more versatile and can handle complex geometries, boundary conditions, and non-linearities. Common methods include; Finite Difference Methods (FDM), Finite Element Methods (FEM), Finite Volume Methods (FVM),

CHANDAN.M **PES1UG22EC059**
ANIRUDH KULKARNI **PES1UG22EC035**

Spectral Methods, etc. CPUs struggle with solving PDEs efficiently due to:

- **High computational demand:** Solving PDEs often requires significant floating-point operations, which can be slow on CPUs.
- **Memory bandwidth limitations:** High-resolution simulations demand substantial data movement, which CPUs may not handle efficiently.
- **Parallelism limitations:** CPUs are not optimized for the fine-grained parallelism required by many numerical methods.

Hardware accelerators can aid in solving PDEs by; Providing specialized computation units which are optimized for specific operations, such as matrix multiplications and stencil computations. Reducing computation time by parallelizing tasks and minimizing data movement. Lowering power consumption through efficient use of resources.

### Numerical Methods for Solving PDEs

- **Finite Difference Method (FDM)**: Approximates derivatives with finite differences using discretized spatial and temporal domains.
- **Jacobi Method**: Iterative method splitting matrix A into diagonal (D) and non-diagonal (R) parts, leveraging data reuse.
- **Gauss-Seidel Method**: Uses the latest available values for faster convergence and reduced memory usage but introduces data dependencies.
- **Checkerboard Method**: Divides grid nodes into two parallelizable groups.
- **Hybrid Iteration Method**: Balances convergence speed and parallelism by using the latest values from top points.

### Hardware Design Considerations

1. **Processing Elements (PEs)**: Design custom PEs optimized for the arithmetic operations required by the PDE solver.
2. **Pipelining**: Implement deep pipelining to ensure continuous data flow and maximize throughput.
3. **Parallelism**: Exploit parallelism by designing multiple PEs to update different grid points simultaneously.
4. **Memory Interface**: Design a high-bandwidth memory interface to ensure efficient data transfer between FPGA and external memory.
5. **Synchronization**: Implement synchronization mechanisms to ensure consistent updates across all Pes.

## 4. FDMAX ARCHITECTURE

FDMAX is a flexible accelerator architecture designed to efficiently support the Finite Difference Method (FDM) for various types of PDEs. It features a reconfigurable Processing Element (PE) array that can adapt to different grid sizes and equations. The architecture optimizes data reuse and minimizes interconnection overhead, making it highly efficient in terms of performance and energy consumption.

### Key Features

- **Reconfigurable PE Design:** Supports different PDEs by adjusting to various computation patterns.
- **Elastic Accelerator Architecture:** Adapts to different PDEs through dynamic subarray configuration.
- **Memory Hierarchies:** Optimized for minimizing data access latency and energy consumption.

### Performance and Efficiency

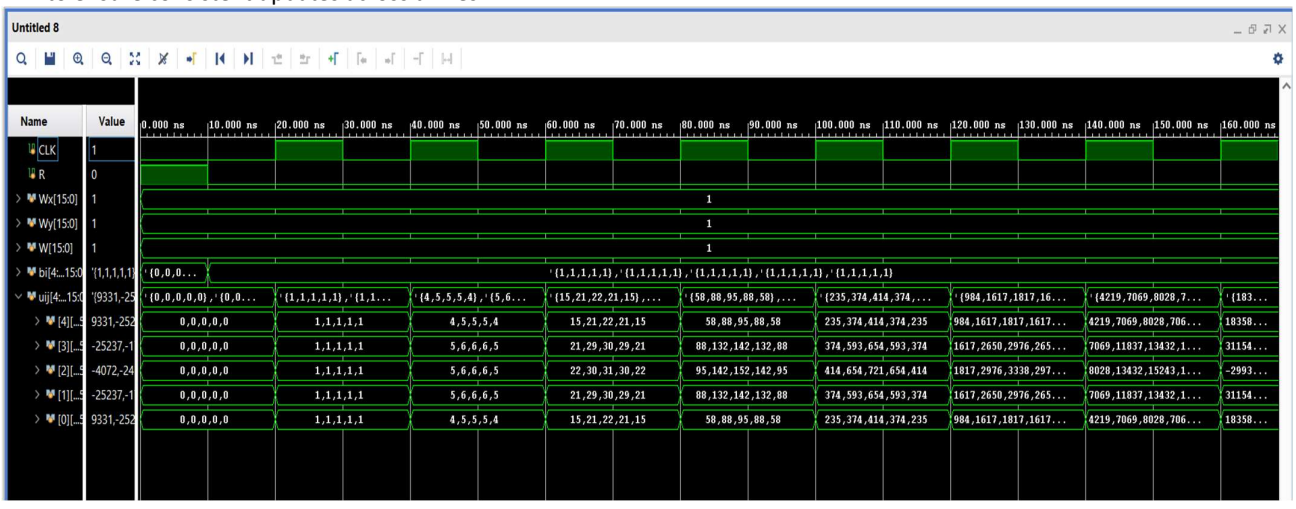
FDMAX has shown significant performance improvements and energy savings compared to traditional CPU and GPU implementations. It achieves an average of 1189× speedup with 1123× energy reduction over an Intel Xeon CPU and 4.9× speedup with 6.3× energy reduction over an NVIDIA RTX3090 GPU.

## 5. Implementation in VIVADO

In this implementation, a 5x5 Processing Element (PE) array is designed using Verilog, where each PE functions as a core computational unit for iterative numerical methods, such as the Jacobi or Gauss-Seidel methods, commonly employed in solving partial differential equations (PDEs). The design integrates a network of `element` modules, with each PE receiving inputs from its adjacent elements, weighted values (`Wx`, `Wy`, `W`), and a bias term (`bi`). The interconnected structure of the PEs facilitates data exchange between neighboring elements, enabling the iterative refinement of solutions across the grid. This modular approach not only allows for efficient computation on FPGA or ASIC platforms but also supports scalability, making it suitable for high-performance applications such as real-time PDE solvers, signal processing, and other scientific computations. The systematic layout of the PE array ensures that the computational workload is evenly distributed, optimizing the overall processing efficiency of the system.

**CODE: -** [Link]

OUTPUT:



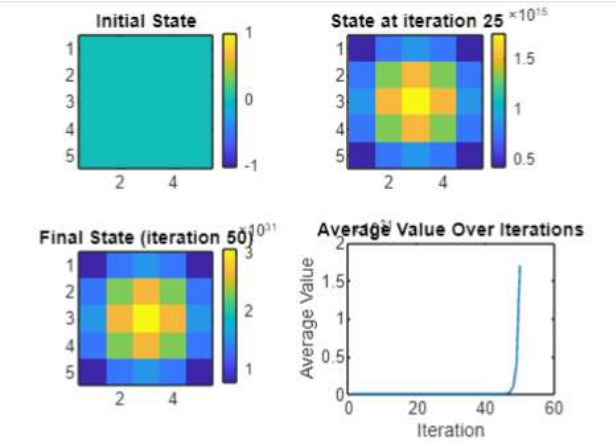

CHANDAN.M **PES1UG22EC059**
ANIRUDH KULKARNI **PES1UG22EC035**

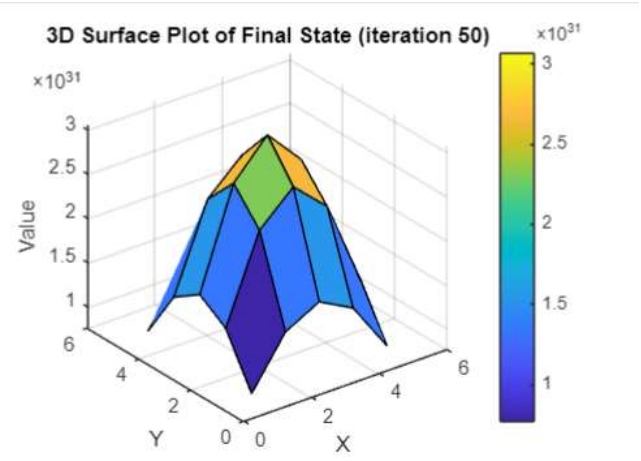
## 6. MATLAB implementation and verification

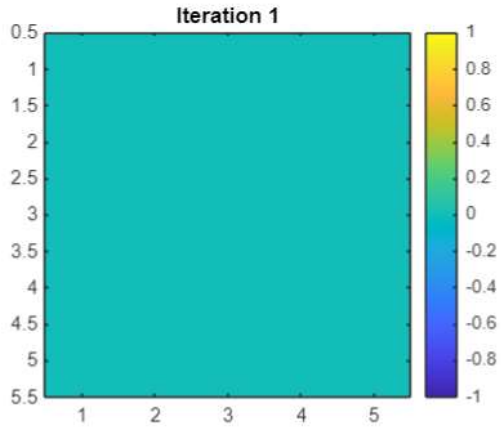
In our verification process, we utilized MATLAB to ensure the accuracy of the Verilog code developed for solving partial differential equations (PDEs) using the FDMAX architecture. The MATLAB simulation involved initializing a 5x5 grid and iterating through 50 cycles to update the grid values based on the finite difference method. The MATLAB code calculates the grid updates by incorporating boundary conditions and weights, and it prints the state at each iteration for comparison with the Verilog implementation. This approach facilitates the validation of the Verilog code by providing a reliable reference for the computed PDE solutions. The final state of the simulation is visualized through a heatmap, offering a clear comparison between the MATLAB and Verilog results.

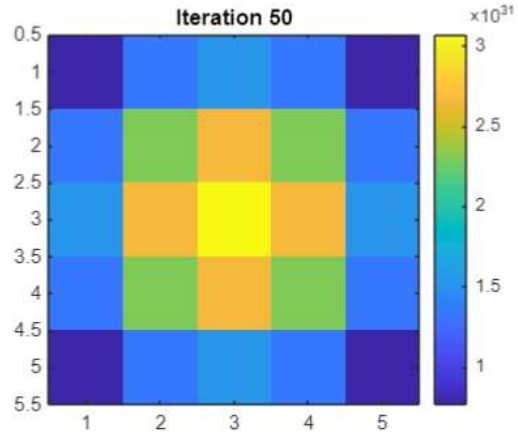
**CODE:-** Link

OUTPUT:




Number of iterations for 5*5 matrix as specified in the above MATLAB code:

```
Iteration 2
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1
Iteration 3
     4     5     5     5     4
     5     6     6     6     5
     5     6     6     6     5
     5     6     6     6     5
     4     5     5     5     4
Iteration 4
    15    21    22    21    15
    21    29    30    29    21
    22    30    31    30    22
    21    29    30    29    21
    15    21    22    21    15
Iteration 5
    58    88    95    88    58
    88   132   142   132    88
    95   142   152   142    95
    88   132   142   132    88
    58    88    95    88    58
Iteration 6
   235   374   414   374   235
   374   593   654   593   374
   414   654   721   654   414
   374   593   654   593   374
   235   374   414   374   235
```

## 7. Conclusion

We have successfully implemented the FDMAX architecture on Vivado showcasing its versatility and effectiveness in different hardware design environments. The implementation was meticulously verified using MATLAB simulations, ensuring that the results from the hardware matched the expected outcomes. This comprehensive approach to design and verification highlights the robustness of the FDMAX architecture in solving partial differential equations (PDEs) efficiently. The FDMAX architecture, in particular, leverages the finite difference method to solve PDEs, offering a scalable solution that can be adapted to various problem sizes and complexities. Its implementation in hardware allows for real-time processing, making it suitable for applications that require rapid and accurate solutions.

## 8. References:


- Jiajun Li, Yuxuan Zhang, Hao Zheng, and Ke Wang. 2023. FDMAX: An Elastic Accelerator Architecture for Solving Partial Differential Equations. In Proceedings of the 50th Annual International Symposium on Computer Architecture (ISCA '23). Association for Computing Machinery, New York, NY, USA, Article 48, 1–12. https://doi.org/10.1145/3579371.3589083