Accelerating Electromagnetic Simulations: A Hardware Emulation Approach

M. Tarek Ibn Ziad*, Yousra Alkabani*, M. Watheq El-Kharashi*, Khaled Salah, and Mohamed AbdelSalam****

*Computer and Systems Engineering Department, Ain Shams University, Cairo, Egypt
**Mentor Graphics, Cairo, Egypt

Email: {mohamed.tarek, yousra.alkabani, watheq.elkharashi}@eng.asu.edu.eg
{khaled_mohamed, mohamed_abdelsalam}@mentor.com

ICECS'15, Cairo, Egypt

Outline

- Introduction
- Finite Element Method (FEM)
- Jacobi Over-Relaxation (JOR)
- Hardware Implementation
- Experimental Setup
- Results and Analysis
- Conclusion and Future Work

Outline

- Introduction
- Finite Element Method (FEM)
- Jacobi Over-Relaxation (JOR)
- Hardware Implementation
- Experimental Setup
- Results and Analysis
- Conclusion and Future Work

Introduction

- Electromagnetic (EM) simulations are the cornerstone in the design process of several real applications.
- Numerical methods (ex: FEM) require solving millions of simultaneous equations.
- The solver part in EM simulations represents a serious bottleneck on traditional CPUs.
- FPGAs are limited by memory and area constraints.
- Emulation technology provides a solution to the memory and area constraints encountered by FPGAs.

Contributions

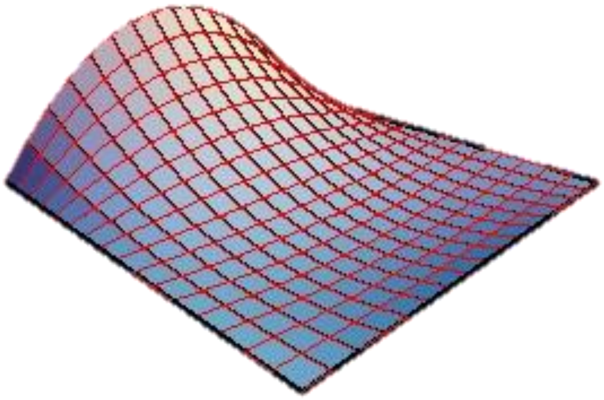
- Proposing an efficient architecture for solving the sparse linear systems arising in FEM formulations based on the Jacobi over-relaxation (JOR) method.
- Optimizing the design by making use of the properties of the FEM coefficients matrix.
- Showing the logic utilization and timing results of implementing the proposed architecture on a commercial hardware emulation platform.

Outline

- Introduction
- Finite Element Method (FEM)
- Jacobi Over-Relaxation (JOR)
- Hardware Implementation
- Experimental Setup
- Results and Analysis
- Conclusion and Future Work

Finite Element Method (FEM)

- Basic procedures of using FEM:
  - ✓ Discretizing the domain into finite elements.
  - ✓ Calculating elemental matrices.
  - ✓ Assembling the elemental matrices to form a global linear system.
  - ✓ **Solving the sparse linear system.**
  - ✓ Post-processing the results.



Outline

- Introduction
- Finite Element Method (FEM)
- Jacobi Over-Relaxation (JOR)
- Hardware Implementation
- Experimental Setup
- Results and Analysis
- Conclusion and Future Work

The Jacobi Over-Relaxation (JOR) Algorithm

- Simple SLS:

$$\begin{cases} 5x_1 + x_2 = 6 \\ x_1 + 5x_2 + 2x_3 = 8 \\ 2x_2 + 5x_3 + x_4 = 8 \\ 2x_3 + 5x_4 = 7 \end{cases}$$

- Matrix form:

$$\underbrace{\begin{bmatrix} 5 & 1 & 0 & 0 \\ 1 & 5 & 2 & 0 \\ 0 & 2 & 5 & 1 \\ 0 & 0 & 2 & 5 \end{bmatrix}}_{\boldsymbol{A}} \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}}_{\boldsymbol{x}} = \underbrace{\begin{bmatrix} 6 \\ 8 \\ 8 \\ 7 \end{bmatrix}}_{\boldsymbol{b}}$$

- Solution:

$$x_i^{(t+1)} = (1 - \omega) * x_i^{(t)} + \frac{\omega}{a_{ii}} \left(b_i - \sum_{j=1, j \neq i}^{j=n} a_{ij} x_j^{(t)} \right)$$

t is the current iteration number

ω is the relaxation parameter [0, 1]

The Jacobi Over-Relaxation (JOR) Algorithm (2)

- Simple SLS:

$$\begin{cases} 5x_1 + x_2 = 6 \\ x_1 + 5x_2 + 2x_3 = 8 \\ 2x_2 + 5x_3 + x_4 = 8 \\ 2x_3 + 5x_4 = 7 \end{cases}$$

- Matrix form:

$$\begin{bmatrix} 5 & 1 & 0 & 0 \\ 1 & 5 & 2 & 0 \\ 0 & 2 & 5 & 1 \\ 0 & 0 & 2 & 5 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 6 \\ 8 \\ 8 \\ 7 \end{bmatrix}$$

- First iteration: $(\omega = 0.5)$

$$\begin{cases} x_1 = (1 - \omega) * 0 + \frac{\omega}{5}(6 - 0) = 0.6 \\ x_2 = (1 - \omega) * 0 + \frac{\omega}{5}(8 - 0) = 0.8 \\ x_3 = (1 - \omega) * 0 + \frac{\omega}{5}(8 - 0) = 0.8 \\ x_4 = (1 - \omega) * 0 + \frac{\omega}{5}(7 - 0) = 0.7 \end{cases}$$

The Jacobi Over-Relaxation (JOR) Algorithm (3)

- Simple SLS:

$$\begin{cases} 5x_1 + x_2 = 6 \\ x_1 + 5x_2 + 2x_3 = 8 \\ 2x_2 + 5x_3 + x_4 = 8 \\ 2x_3 + 5x_4 = 7 \end{cases}$$

- Matrix form:

$$\begin{bmatrix} 5 & 1 & 0 & 0 \\ 1 & 5 & 2 & 0 \\ 0 & 2 & 5 & 1 \\ 0 & 0 & 2 & 5 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 6 \\ 8 \\ 8 \\ 7 \end{bmatrix}$$

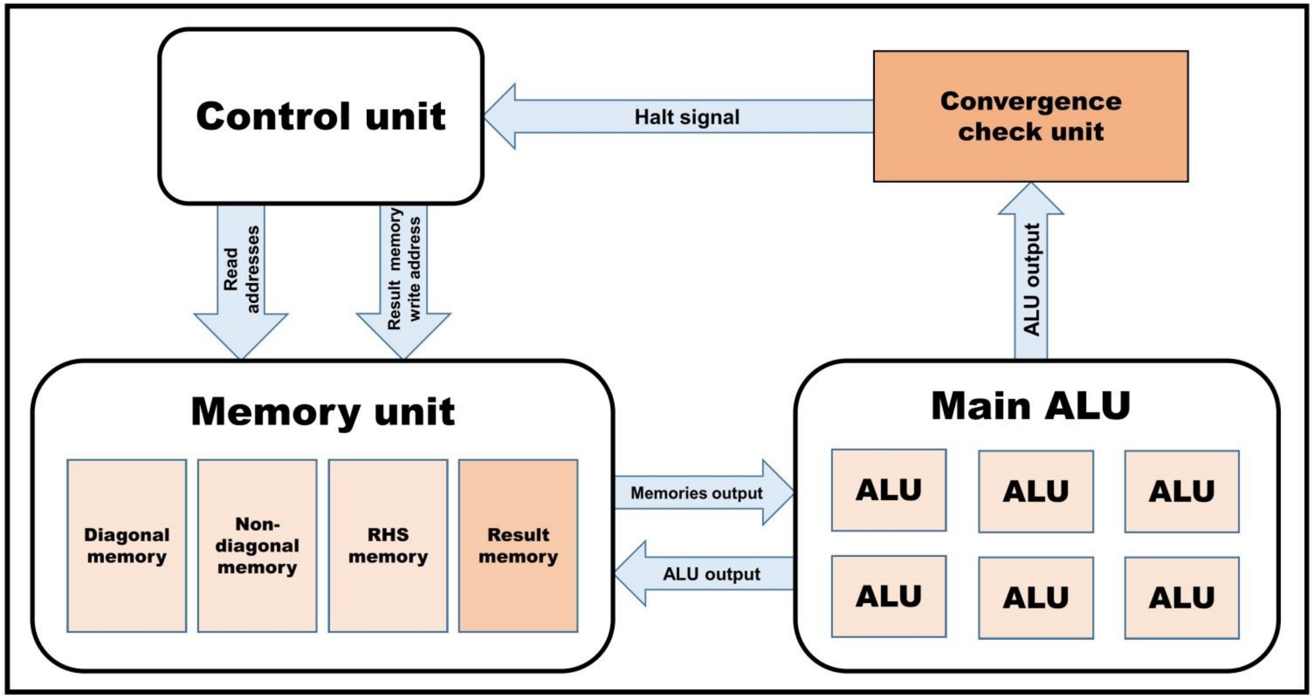
- After t iterations:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \underset{t=1}{\begin{bmatrix} 0.6 \\ 0.8 \\ 0.8 \\ 0.7 \end{bmatrix}} \rightarrow \underset{t=2}{\begin{bmatrix} 0.82 \\ 0.98 \\ 0.97 \\ 0.89 \end{bmatrix}} \rightarrow \underset{t=3}{\begin{bmatrix} 0.912 \\ 1.014 \\ 1.000 \\ 0.951 \end{bmatrix}} - - - \rightarrow \begin{bmatrix} 1.0 \\ 1.0 \\ 1.0 \\ 1.0 \end{bmatrix}$$

Outline

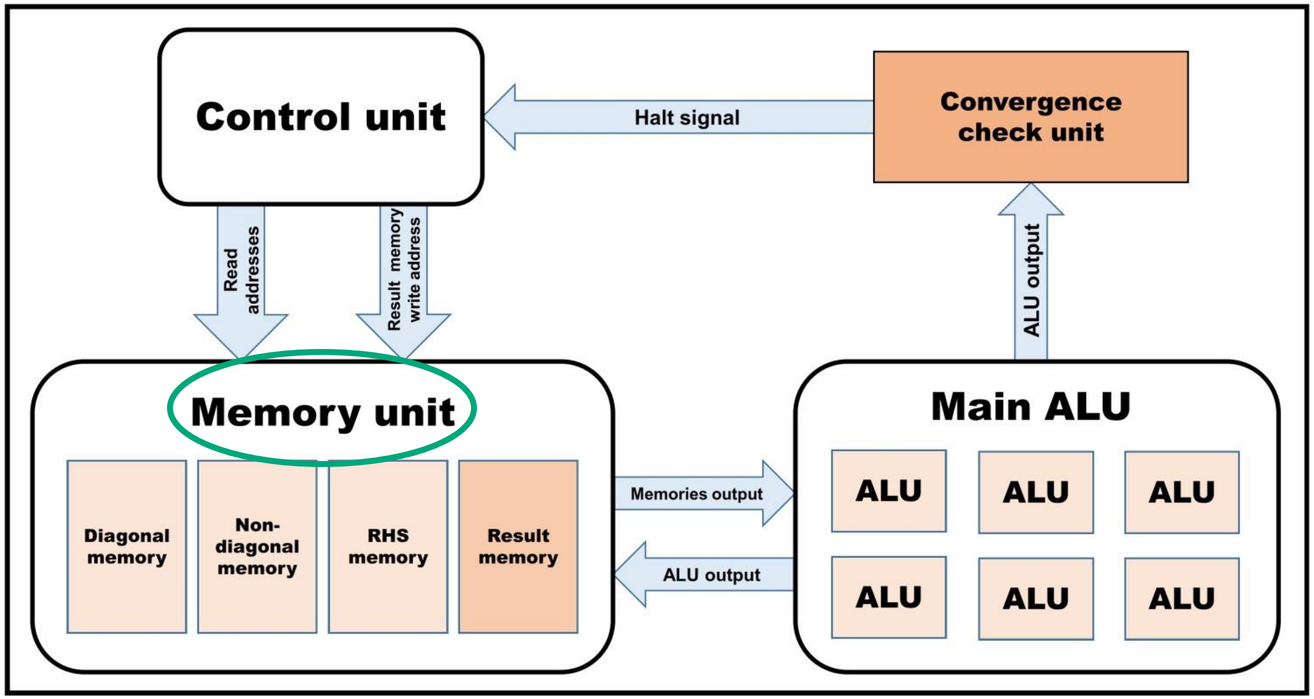
- Introduction
- Finite Element Method (FEM)
- Jacobi Over-Relaxation (JOR)
- Hardware Implementation
- Experimental Setup
- Results and Analysis
- Conclusion and Future Work

Hardware Implementation



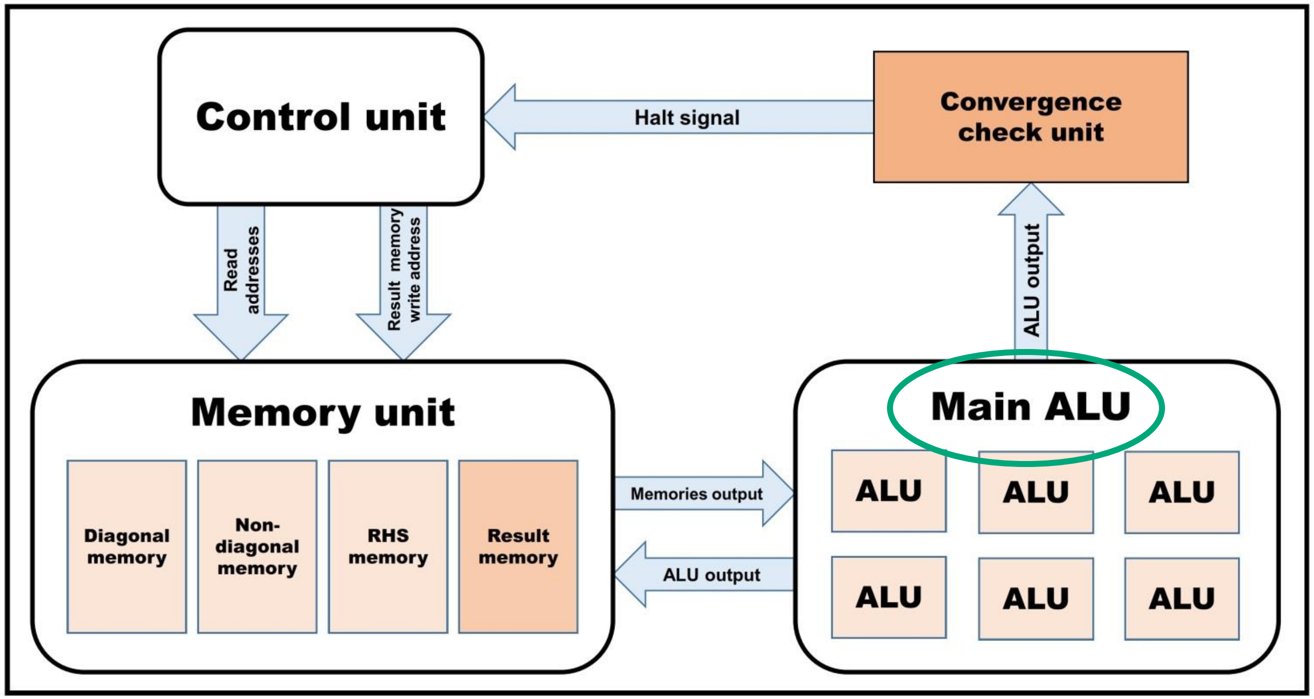
- Our architecture includes four main components; the memory unit, the main ALU, the convergence check unit, and the control unit.

Hardware Implementation - Memory



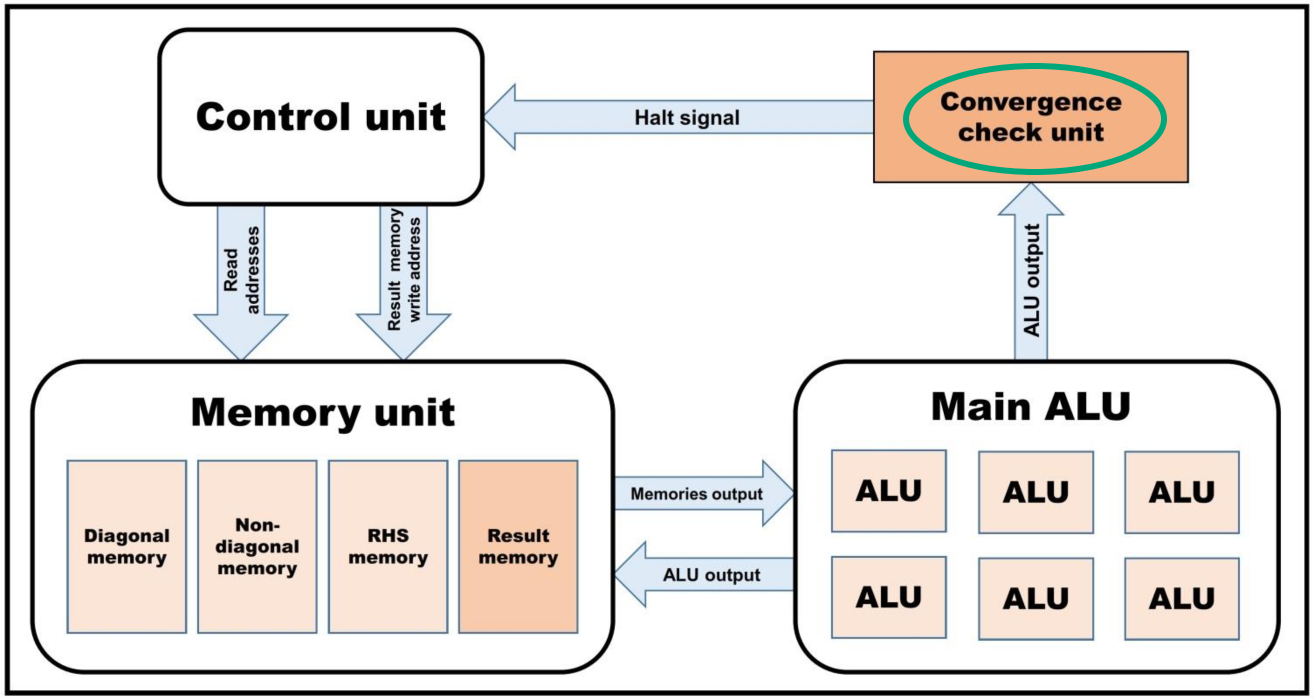
- The memory unit consists of four separate memories; the diagonal memory, the non-diagonal memory, the RHS memory, and the result memory.
- Each memory row contains a whole cluster of data, not the elements of a single A row.

Hardware Implementation - ALU



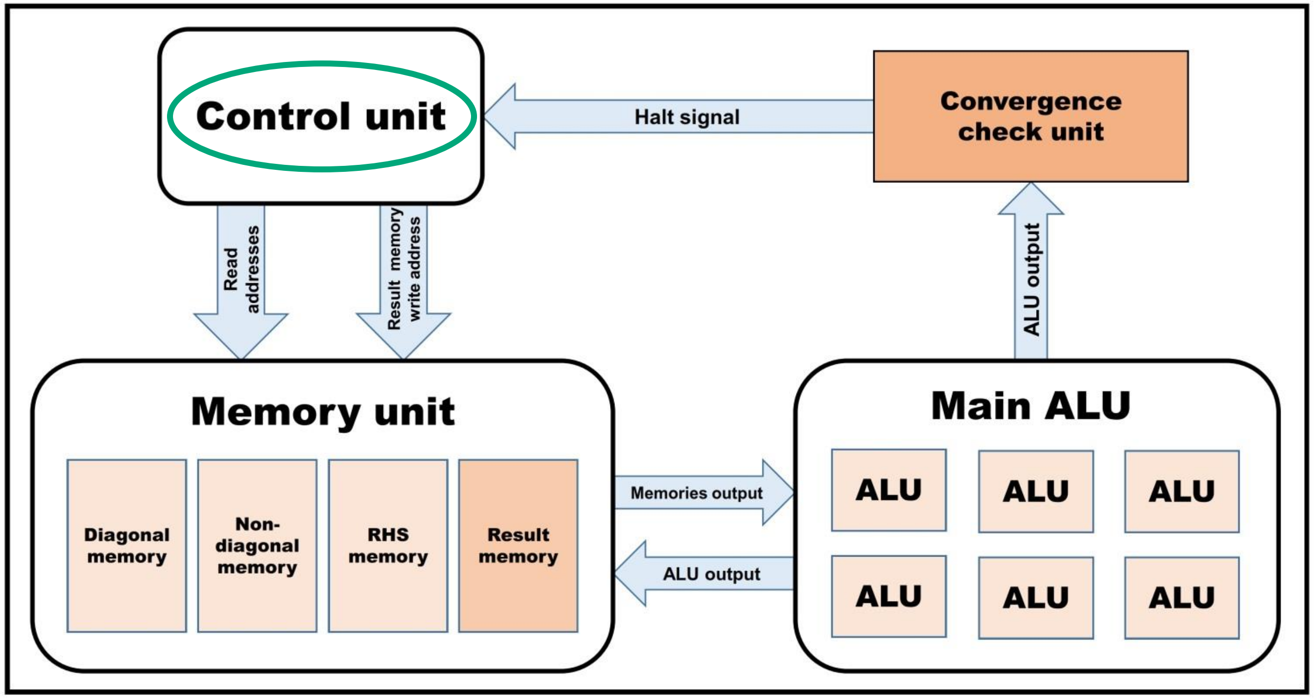
- The main ALU contains a number of independent ALUs, equaling the number of clusters, c.
- All ALUs are identical and are responsible for all arithmetic operations performed on data.

Hardware Implementation – Convergence Check Unit



- Convergence check unit decides when the flow should terminate.
- The termination criterion is determined based on a pre-defined FP value, 10^{-6}, representing the accepted error tolerance.

Hardware Implementation – Control Unit



- The control unit is responsible for synchronizing all memories with each ALU and controlling the convergence check unit.

Outline

- Introduction
- Finite Element Method (FEM)
- Jacobi Over-Relaxation (JOR)
- Hardware Implementation
- Experimental Setup
- Results and Analysis
- Conclusion and Future Work

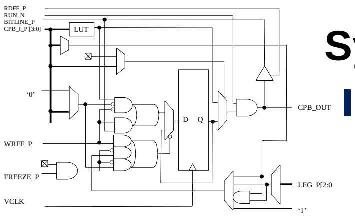
Experimental Setup

RTL Design

```
module JOR(clk, reset, ALU_out, halt);

    input wire clk, reset;

    output wire [447: 0] ALU_out;
    output halt;

    always @ (posedge clk)
    begin
        //Code
    end

endmodule
```

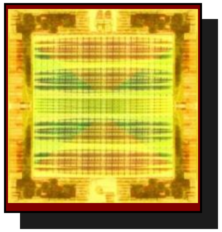
Compiler

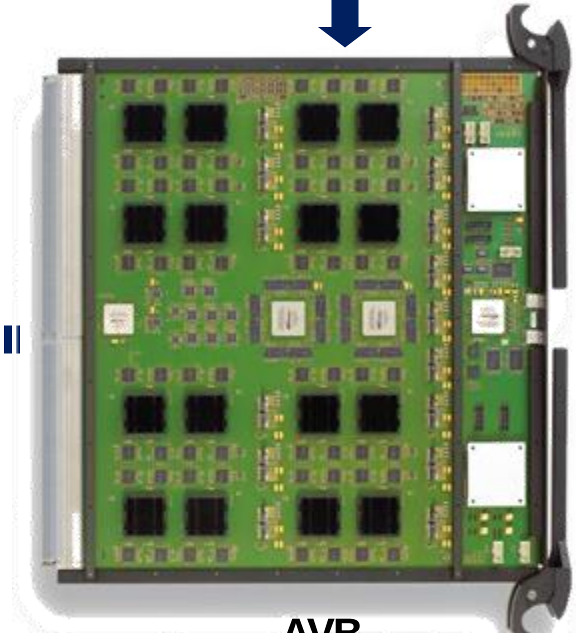
ASIC Gates



Synthesizer

Crystal chip



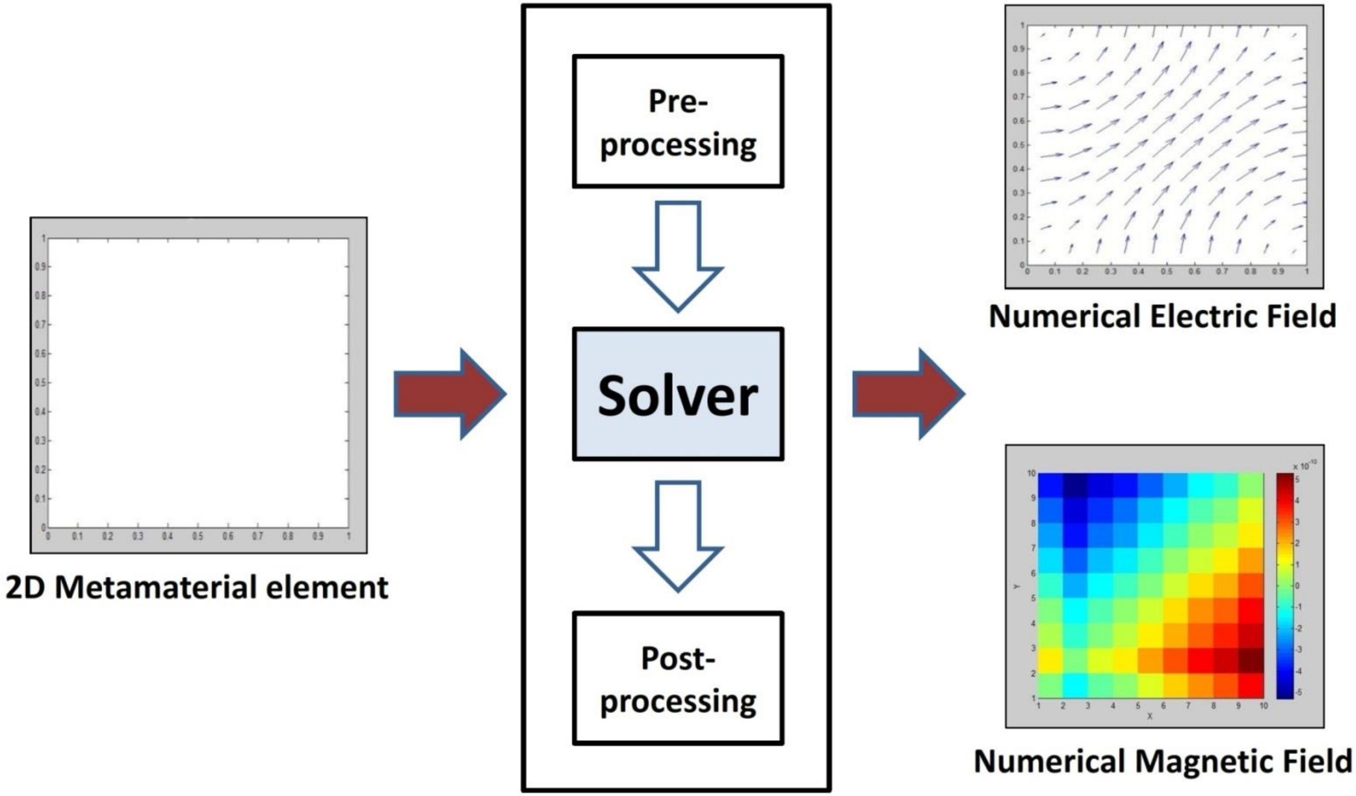


AVB

Outline

- Introduction
- Finite Element Method (FEM)
- Jacobi Over-Relaxation (JOR)
- Hardware Implementation
- Experimental Setup
- Results and Analysis
- Conclusion and Future Work

Results and Analysis – EM Simulator



General steps included in the selected EM simulator.

Results and Analysis – Resource Utilization

	Proposed JOR design test cases			
Number of equations	420	11,100	44,700	2,002,000
Number of ALUs	14	74	149	1,000
Frequency (MHz)	2.00	2.00	2.00	1.75
Number of iterations	5	5	5	5
Number of LUTs	164,449	770,432	1,527,869	10,122,146
Number of flip-flops	40,833	167,751	326,382	2,126,259
Memory usage (KB)	152.2	512	1,630	40,128

- The operating frequency, resource utilization, and memory capacity of our JOR design with single-precision FP accuracy for different test cases.

Results and Analysis – Timing Performance

Number of equations	Proposed JOR design	MatJOR		ALGLIB	
	Time (msec)	Time (msec)	Speed-up	Time (msec)	Speed-up
420	0.104	0.345	3.32	0.350	3.37
11,100	0.464	9.292	20.03	3.001	6.47
44,700	0.914	39.581	43.31	12.00	13.13
2,002,000	4.580	2393.085	522.51	847.049	184.95

- The timing performance of our JOR design is evaluated by comparing the needed time to solve a given number of equations using our JOR design against two software solvers; MatJOR and ALGLIB on a 2.00 GHz Core i7-2630QM CPU.

Results and Analysis – Comparison Vs. Previous Work

Number of equations	Proposed JOR design Time (msec)	Proposed JOR design LUTs/FFs (×1000)	Jacobi design in [5] Time (msec)	Jacobi design in [5] LUTs/FFs (×1000)	Speed-up	Area overhead (%)
420	0.104	164/40	0.319	106/18	3.07	65%
11,100	0.464	770/167	1.721	535/85	3.71	51%
44,700	0.914	1,527/326	3.793	1,071/168	4.15	49%
2,002,000	4.580	10,122/2,126	24.040	7,158/1,115	5.25	48%

- Area and Time comparisons against the Jacobi design in [5] on the same hardware emulation platform.
- Area overhead due to the three more FP modules in the JOR design compared to the one in [5].
- Speed-ups up to 5.25x due to the higher convergence rate of the JOR method.

Outline

- Introduction
- Finite Element Method (FEM)
- Jacobi Over-Relaxation (JOR)
- Hardware Implementation
- Experimental Setup
- Results and Analysis
- Conclusion and Future Work

Conclusion and Future Work

- We presented a FP architecture for solving SLS generated from FEM using the JOR method.
- We implemented our JOR hardware solver on a physical hardware emulation platform.
- Future work includes evaluating the efficiency of our hardware solver against the latest GPU solvers.
- We aim to investigate more complicated methods in order to build the first electromagnetic field emulator in the world.

Thank you