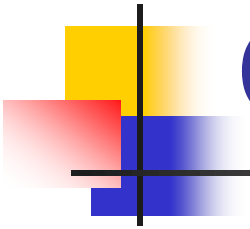
On Hardware Solution of Dense Linear Systems via Gauss-Jordan Elimination

M. Tarek Ibn Ziad*, Yousra Alkabani*, and M. Watheq El-Kharashi**

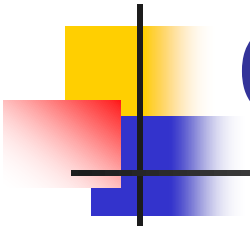
*Computer and Systems Engineering Department
Ain Shams University, Cairo, Egypt
**Department of Electrical and Computer Engineering
University of Victoria, Victoria, Canada

Email: {mohamed.tarek, yousra.alkabani}@eng.asu.edu.eg
watheq@engr.uvic.ca



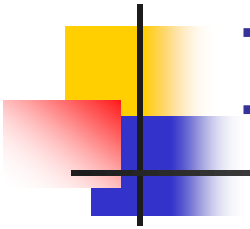
Outline

- Introduction
- The Gauss-Jordan Elimination Algorithm
- Hardware Implementation
- Experiments and Analysis
- Conclusion



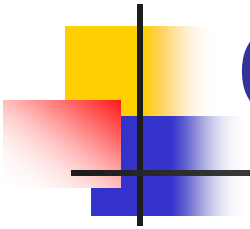
Outline

- Introduction
- The Gauss-Jordan Elimination Algorithm
- Hardware Implementation
- Experiments and Analysis
- Conclusion



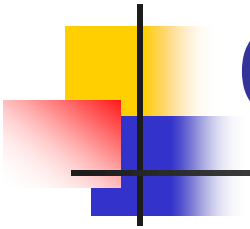
Introduction

- Solving systems of linear equations represents the core operation in a wide variety of applications in fundamental sciences.
- One of the most widely used methods for solving dense linear systems (DLS) is the Gauss-Jordan Elimination (GJE) method.
- Current FPGA implementations of the GJE on FPGAs only care about area and throughput. There are no energy-aware hardware implementations.



Contributions

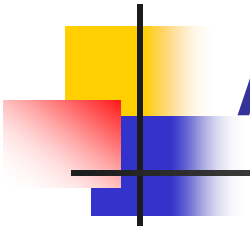
- A pipelined architecture for solving DLS using the GJE algorithm with single-precision floating-point (FP) accuracy.
- A detailed experimental analysis of the design logic utilization, time performance, and power consumption.
- A performance comparison and evaluation of our proposed technique against similar hardware implementations with respect to time and area.



Outline

- Introduction
- The Gauss-Jordan Elimination Algorithm
- Hardware Implementation
- Experiments and Analysis
- Conclusion

The Gauss-Jordan Elimination Algorithm



- Inputs: matrix, A and RHS vector, b.
- Output: solution vector, x.
- Steps:
  - ✓ Pivot element location.
  - ✓ Row exchange.
  - ✓ Current row normalization.
  - ✓ Row elimination.

The Gauss-Jordan Elimination Algorithm - Example

- Simple DLS:

$$\begin{cases} 4x + 4y + 4z = 20 \\ 2x + 3y + 5z = 8 \\ 4x + 5z = 2 \end{cases}$$

- Augmented matrix:

$$\begin{bmatrix} 4 & 4 & 4 & 20 \\ 2 & 3 & 5 & 8 \\ 4 & 0 & 5 & 2 \end{bmatrix}$$

- 1st row normalization:

$$\begin{bmatrix} 1 & 1 & 1 & 5 \\ 2 & 3 & 5 & 8 \\ 4 & 0 & 5 & 2 \end{bmatrix}$$

- Rows elimination:

$$\begin{bmatrix} 1 & 1 & 1 & 5 \\ 0 & 1 & 3 & -2 \\ 0 & -4 & 1 & -18 \end{bmatrix}$$

The Gauss-Jordan Elimination Algorithm – Example (2)

- Pivot element location:

$$\begin{bmatrix} 1 & 1 & 1 & 5 \\ 0 & 1 & 3 & -2 \\ 0 & -4 & 1 & -18 \end{bmatrix}$$

- Rows elimination:

$$\begin{bmatrix} 1 & 0 & -2 & 7 \\ 0 & 1 & 3 & -2 \\ 0 & 0 & 13 & -26 \end{bmatrix}$$

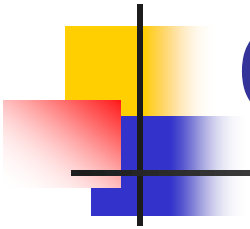
- 3rd row normalization:

$$\begin{bmatrix} 1 & 0 & -2 & 7 \\ 0 & 1 & 3 & -2 \\ 0 & 0 & 1 & -2 \end{bmatrix}$$

- Rows elimination:

$$\begin{bmatrix} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 4 \\ 0 & 0 & 1 & -2 \end{bmatrix}$$

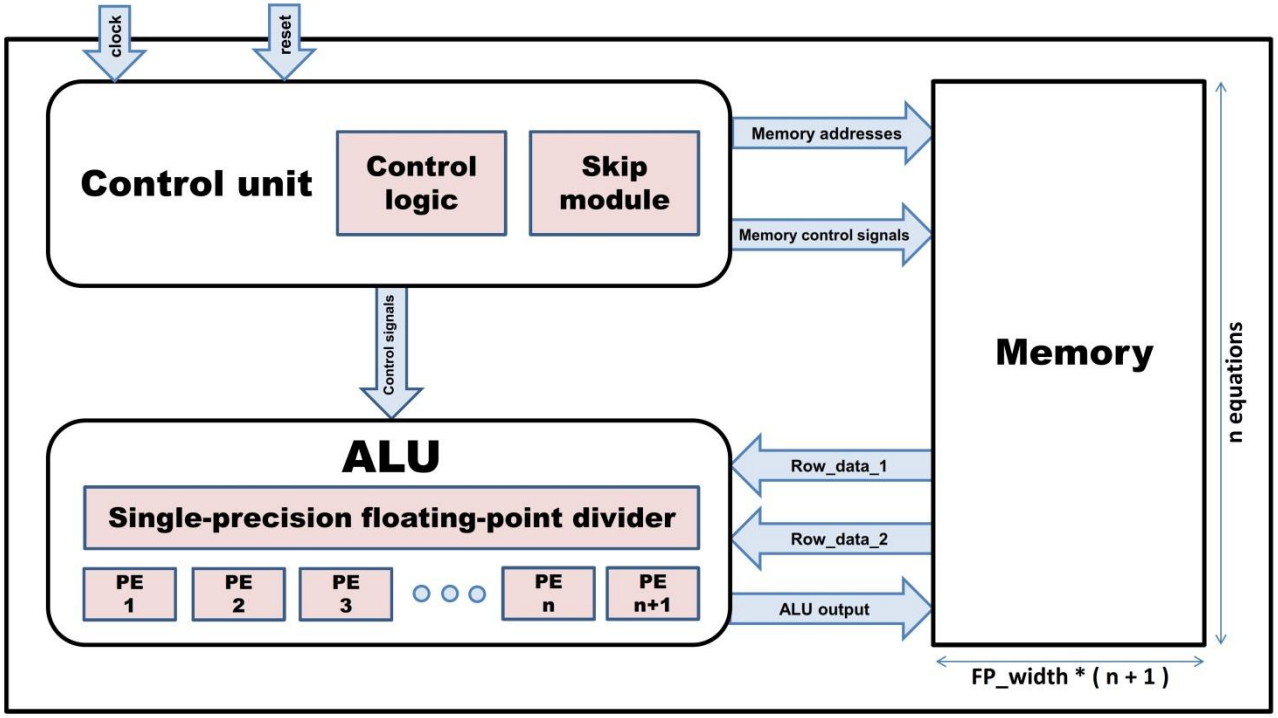
$$x$$



Outline

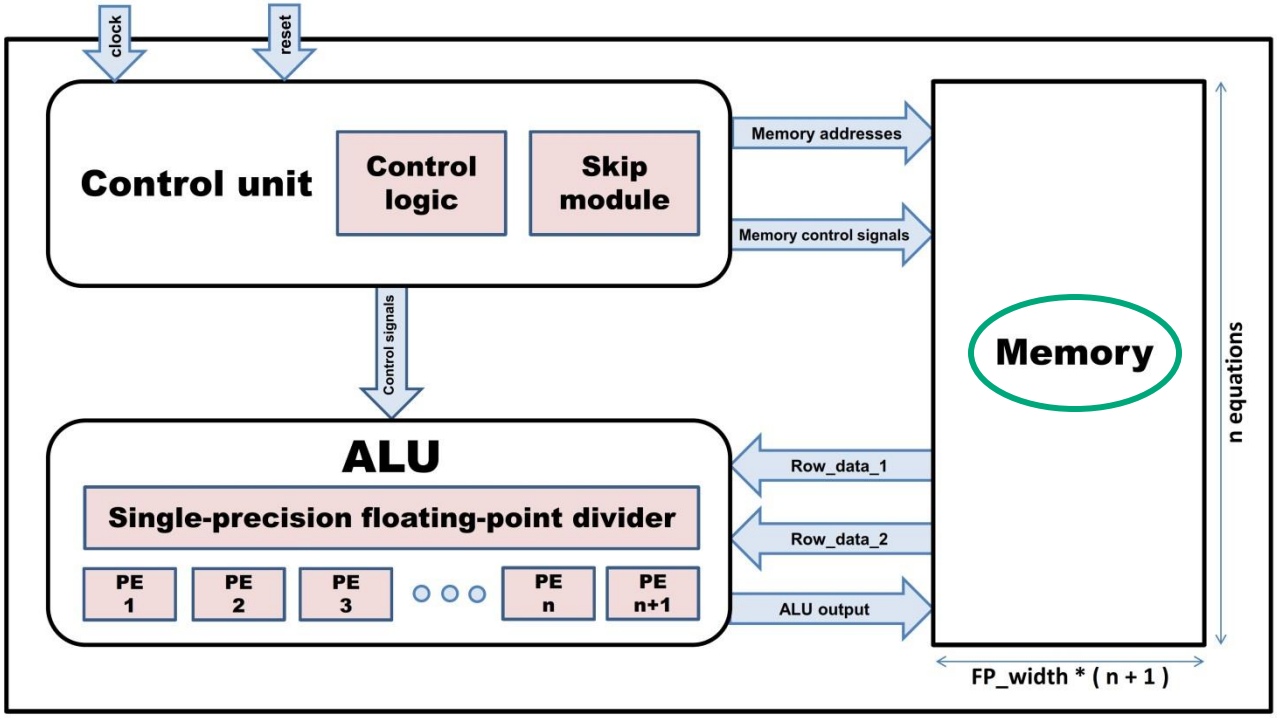
- Introduction
- The Gauss-Jordan Elimination Algorithm
- Hardware Implementation
- Experiments and Analysis
- Conclusion

Hardware Implementation



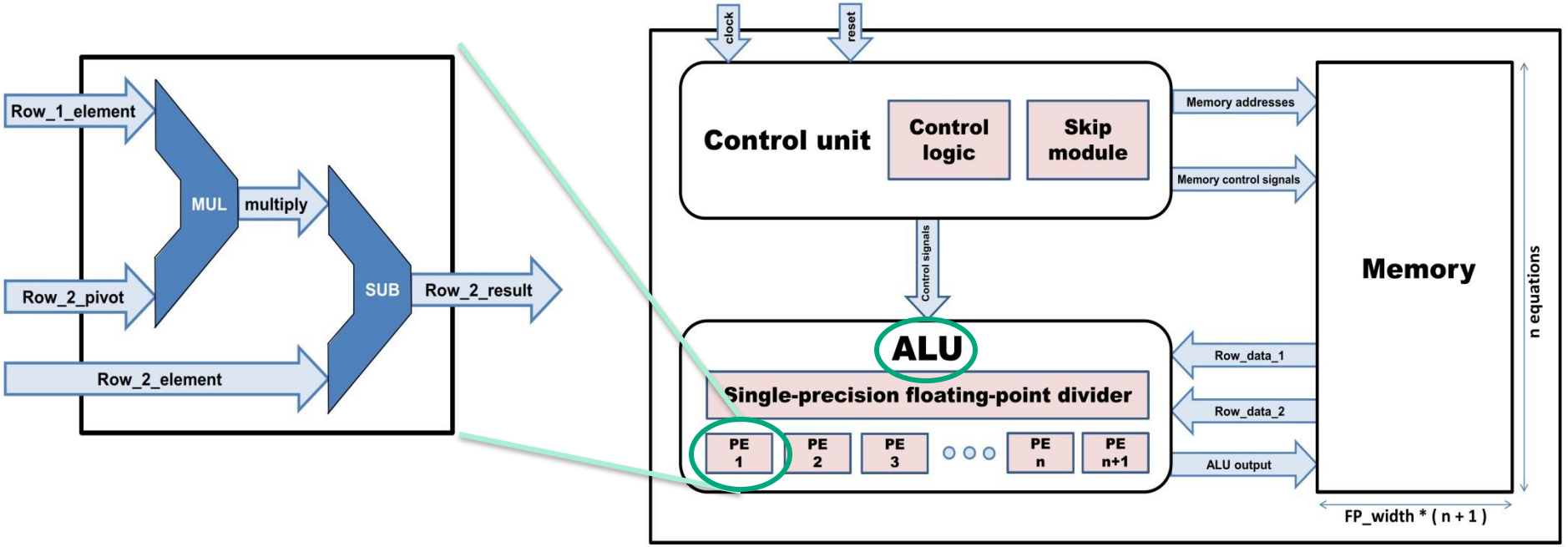
- Our architecture includes three main components; memory, control unit, and ALU.

Hardware Implementation - Memory



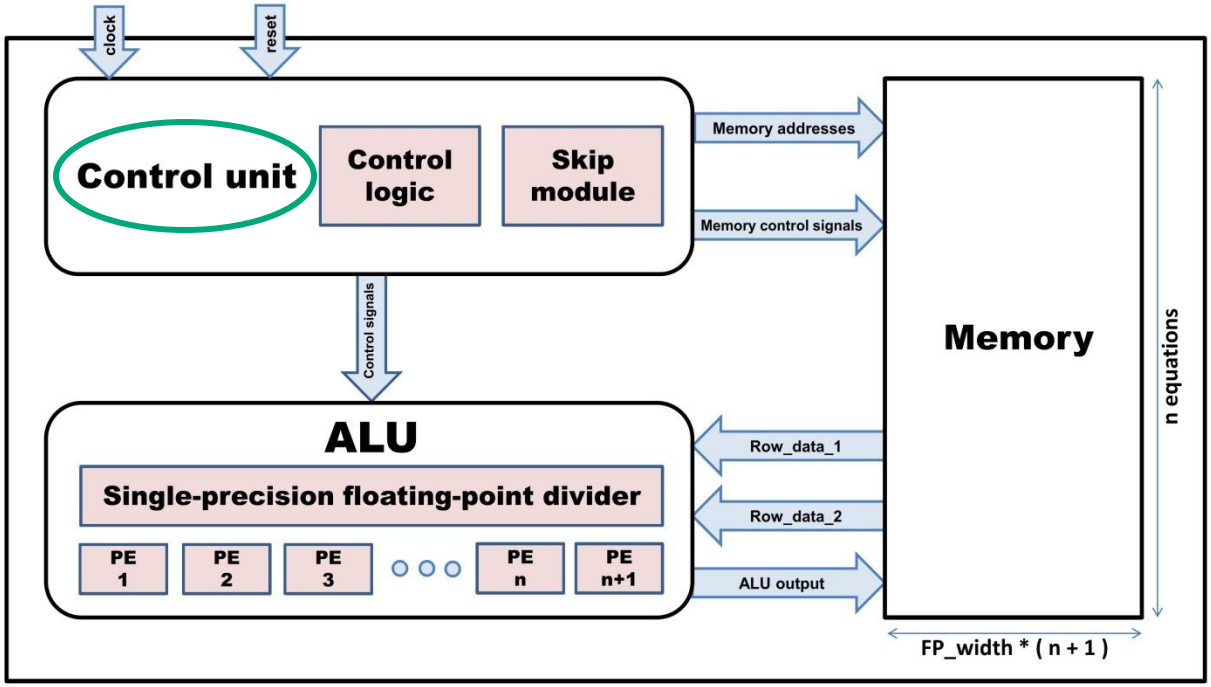
- Our memory unit is a three-port memory with depth equals the number of equations, n.
- Write/read enable signals are used to enable memory in specific times and disable it otherwise to reduce power consumption.

Hardware Implementation - ALU



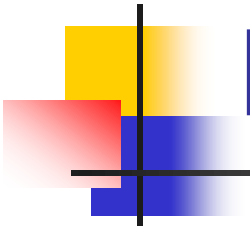
- Our ALU consists of only one single-precision FP divider and n + 1 processing elements (PEs).
- All PEs are identical and are mainly used to perform the row elimination step in parallel.

Hardware Implementation - Control Unit

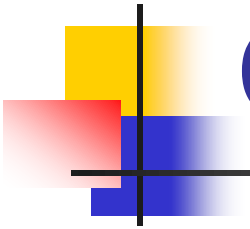


- The control unit consists mainly of two submodules; the control logic, which is responsible for controlling the signals connected to memory and ALU, and the skip module, which skips the normalization step when the pivot element is already 1 and skips the elimination step when the eliminated element is already 0.

Hardware Implementation - Floating-point (FP) Modules



- We utilize single-precision FP modules, generated using FloPoCo, an open source generator of operators written in C++.
- The used FP modules are deeply pipelined in order to obtain the maximum performance.
- Although the FP modules operate on 34 bit operands instead of 32 bit ones, FloPoCo provides simple conversion operators from and to the IEEE-754 standard formats.



Outline

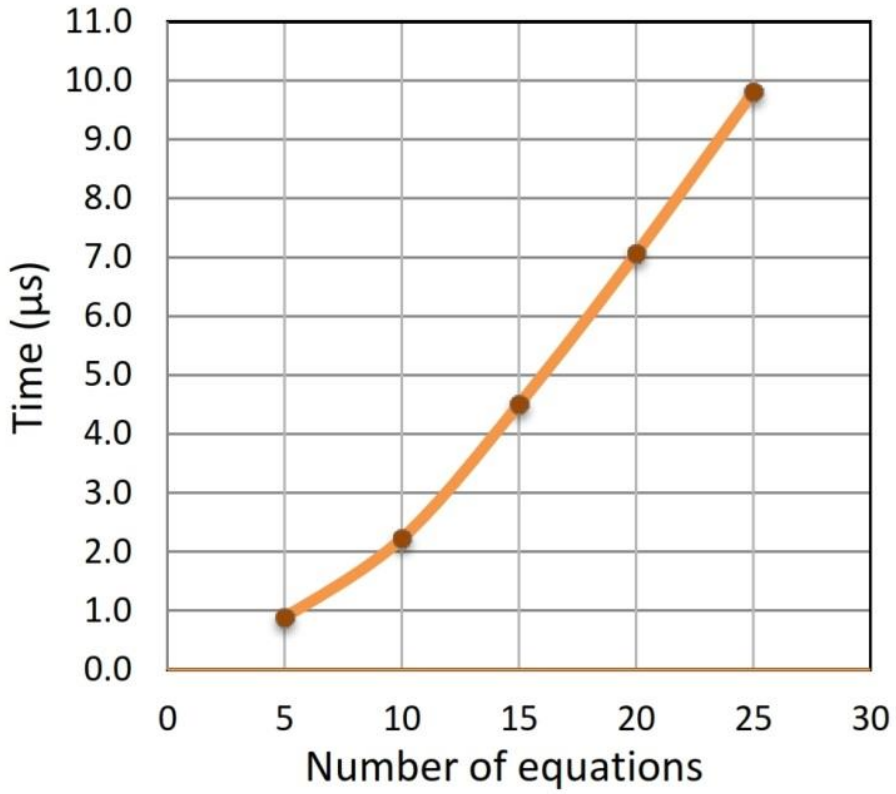
- Introduction
- The Gauss-Jordan Elimination Algorithm
- Hardware Implementation
- Experiments and Analysis
- Conclusion

Experiments and Analysis

No. of equations	5	10	15	20	25
No. of slice registers	3,979	6,313	9,197	11,737	14,262
No. of slice LUTs	6,480	11,022	17,312	23,494	32,773
No. of DSP48Es	24	44	64	84	104
Maximum frequency	198.230	179.880	149.897	141.606	140.183

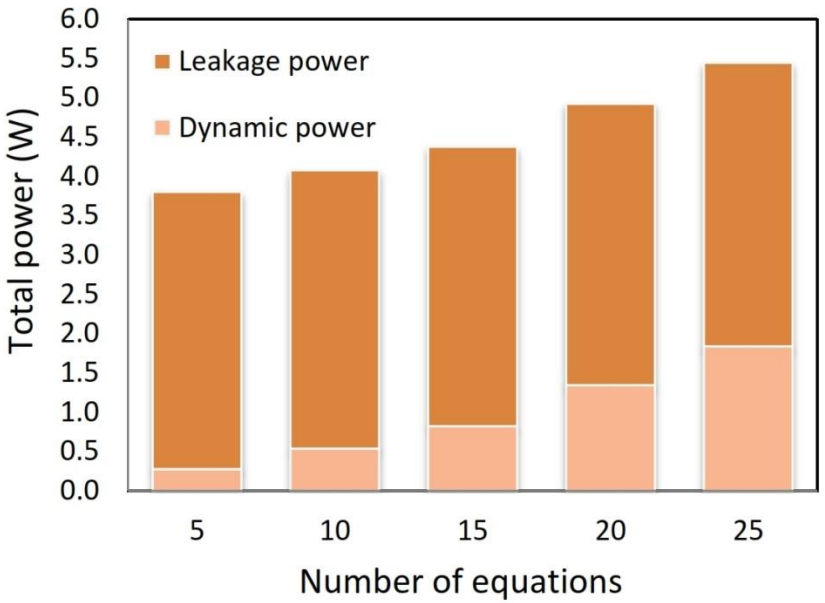
- Hardware resource utilization for our single-precision FP design on Virtex-5 XC5VLX330T FPGA using different test cases.
- It is worth mentioning that we did not optimize the code for a certain FPGA, while FPGA-specific optimizations might yield better area usage.

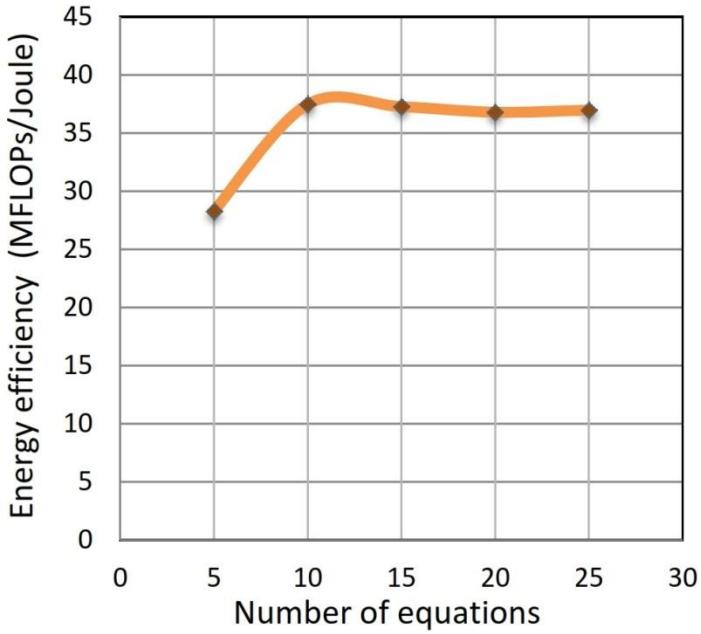
Experiments and Analysis (2)



- Timing results of our GJE implementation on Virtex-5 XC5VLX330T FPGA using different test cases.

Experiments and Analysis (3)



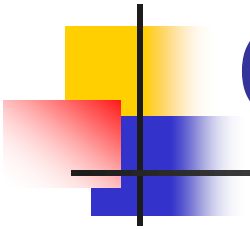


- Power consumption of our GJE implementation on Virtex-5 XC5VLX330T FPGA using different test cases.
- Energy efficiency of our GJE implementation on Virtex-5 XC5VLX330T FPGA using different test cases.

Experiments and Analysis (4)

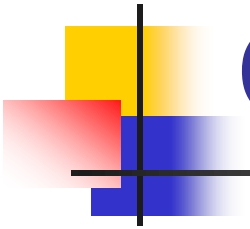
Matrix size	Our design		GJE design [4]	
	4	8	4	8
No. of slice registers	3823	5369	3048	4321
No. of slice LUTs	5820	9184	4476	7396
No. of DSP48Es	20	36	10	10
Number of cycles	136	304	608	1204
Maximum frequency (MHz)	196.142	179.277	263.116	263.116
Execution time (μs)	0.693	1.696	2.311	4.576

- Resources and timing comparisons against the GJE design in [4] using Virtex-5 LX50T FPGA.
- direct comparisons against in terms of energy efficiency are not applicable as the authors in [4] does not offer power analysis of presented implementations.



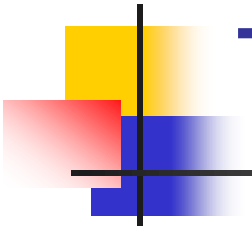
Outline

- Introduction
- The Gauss-Jordan Elimination Algorithm
- Hardware Implementation
- Experiments and Analysis
- Conclusion



Conclusion

- We presented a single-precision FP architecture for solving generic DLS of equations using the GJE algorithm.
- We implemented our design on a Virtex-5 FPGA and discussed detailed experimental results to show time, area, and power costs.
- We aim to investigate the trade-off between obtaining higher solution accuracy by using double and quadruple FP precision and maintaining area and energy efficiency in the future work.



Thank you

FOR QUESTIONS: PLEASE CONTACT
MOHAMED.TAREK@ENG.ASU.EDU.EG