Challenge #17 – Bubble Sort Using a Systolic Array

Course: ECE 410/510

Student: Megha Sai Sumanth Kurra

Introduction

A systolic array is a mesh of interconnected processing units that transfer and compute data in a rhythmic, clock-driven manner. Each processing element (PE) handles basic operations and passes results to its neighboring PE. This structure is especially efficient for predictable and structured computations like sorting or matrix processing.

This challenge explored implementing the bubble sort algorithm using a linear systolic array. Each element performs comparisons with its adjacent unit and swaps values if necessary. After multiple data propagation cycles, the list becomes sorted. The aim was to simulate this sorting method, measure execution performance, and compare it with Python's optimized sorting algorithm.

Methodology

1. Array Structure and Logic:
- Each element in the array corresponds to a processing unit.
- Each unit compares with its neighboring unit and performs conditional swaps.
- Repeating this for N-1 cycles ensures full sorting for N elements.

2. Simulation Approach:
- The systolic array algorithm was written in Python using a list to simulate the PEs.
- Arrays of increasing sizes were generated with random integers.
- Time measurements were taken for each size to evaluate performance.
- A comparative analysis was done against Python's built-in sort to understand the trade-offs.

Results

Sample Run:
Before: [93, 7, 21, 85, 65, 29, 8, 0, 52, 18]
After: [0, 7, 8, 18, 21, 29, 52, 65, 85, 93]

Execution Time Summary:

Array Size	Execution Time
10	0.00005 seconds
100	0.00058 seconds
1000	0.08401 seconds
5000	3.05290 seconds
10000	7.46665 seconds

Discussion

The systolic sorting mechanism emulates data propagation and local interaction within hardware components. It demonstrates how neighboring units can collaboratively sort values without centralized control. As anticipated, the sorting time increases sharply with array size due to the quadratic nature of bubble sort.

Insights:
- Simple to implement and understand, making it a great educational tool.
- Not optimal for large-scale data processing in software environments.
- Performance is inferior to native language sorting functions, which utilize more advanced algorithms and low-level optimizations.

Conclusion

This challenge offered practical exposure to systolic computing principles applied to sorting problems. The hands-on approach solidified the understanding of localized computation and time complexity impact. While unsuitable for high-performance software applications, the systolic model provides a foundation for hardware-accelerated solutions in custom silicon or FPGA designs.

Key Takeaways:
- Modeling systolic behaviors in software helps grasp architectural principles.
- Bubble sort, despite being simple, becomes computationally expensive with scale.
- Algorithm selection is vital when performance and scalability are priorities.