Matrix Multiplication FPGA vs. CPU

Matrix Multiplication

Different algorithms can be used to perform matrix multiplication

- Iterative(naive) algorithm
- Cache behavior
- Strassen algorithm
- Divide-and-conquer algorithm

We will be using the divide-and-conquer algorithm:

- Works for all square matrices with dimensions of the power of 2
- Computes smaller matrices recursively

CPU Hardware Comparison:

Table I
Comparison of Potential (Laptop) CPUs

Name	# of Cores	# of Threads	Base Clock Frequency (GHz)	Boost Clock Frequency (GHz)	TDP (W)
AMD Ryzen 7 5800HS	8	16	2.8	4.4	35
AMD Ryzen 9 5900HS	8	16	3.0	4.6	35
Intel Core I5 7600K	4	4	3.8	4.2	91

FPGA Hardware Comparison:

Table II
Comparison of Potential FPGA Boards
(Modified From [1] Table 6.1)

Name	Total board RAM (Mb)	DSP Slices	Flip Flops	LUTs	Price
Kintex UltraScale KU115	75.9	5, 520	1, 326, 720	663, 360	$6,495.00
Nexys A7-50T	8,150	120	65,200	32,600	N/A
Nexys A7-100T	4.860	240	126,800	63,400	$349.00

Though it is significantly less robust than the Kintex UltraScale KU115 used in [1], we will be using the Nexys A7-100T due to its availability to us as students and much cheaper cost.

Metric Descriptions

n - matrix dimension for a square (n x n)

kernel - threshold where recursion stops and base kernel mulCalc runs. Smaller values = deeper recursion (more overhead). Larger values = shallower recursion.

threads - number of threads the algorithm will use. Higher numbers = more parallel work

iters - number of iterations it repeated the run. Used to average the data and select the best runs

best_ms - best (minimum) elapsed time across iterations for the configuration

gflops - giga floating point operations per second. Higher is better.

References

[1] G. Maan, “Hardware acceleration of matrix multiplication,” Bachelor Thesis, Leiden Institute of Advanced Computer Science, Leiden University, Leiden, The Netherlands, 2019. [Online]. Available: https://theses.liacs.nl/pdf/2018-2019-MaanGC.pdf