Lab 4 Report Yash Shah

CS 133 – 12/6. UID: 405-565-567

Please explain the parallelization and optimization strategies you have applied for each loop in the CNN program (convolution, max pooling, etc) in this lab. Include the pragmas (if any) or code segments you have added to achieve your strategy.

I added parallelization and optimization only in the convolution portion of the CNN. They are as follows:

1. **Pipelining the 'j' loop**: The pragma #pragma ACCEL pipeline is applied before the 'j' loop. This enables different iterations of the loop to be overlapped.
2. **Parallelizing the 'w' loop**: The pragma #pragma ACCEL parallel factor=8 is applied to the 'w' loop. This parallel pragma allows the loop to process 8 iterations concurrently, which increases the computational throughput. It is appropriate here since each iteration of the 'w' loop is independent of others, meaning there are no data dependencies that would prevent parallel execution.

Please incrementally evaluate each parallelization/optimization that you have applied and explain why it improves the performance.

Pipelining the ‘j’ loop increased the performance from the original 10 GFlops to ~11GFlops, and then I had the ‘h’ loop parallized which improved the performance to ~13 GFlops. I then moved the parallel directive to the ‘w’ loop and that dramatically increased my performance from ~13 GFlops to ~50 GFlops with the parallel factor=4. I then increased the parallel factor which brought the execution time from ~50 GFlops to ~90 GFlops.

Please explain how your strategy differs from your Lab 3, and why

Lab 3's GPU-based strategy utilizes CUDA, which involves assigning different parts of the computation to threads within a warp that are executed in a single instruction with multiple threads. This is effective on GPUs where thousands of lightweight threads execute the same instruction on different data elements.

In contrast, Lab 4's FPGA-based strategy uses HLS with Merlin pragmas to exploit parallelism. It involves loop transformations like pipelining and parallelizing with pragmas that instruct the HLS tool to optimize the computation for the FPGA's spatial architecture. FPGAs excel in executing multiple distinct instructions concurrently across different data paths, which is the fundamental difference between the two labs.

The other key difference is that GPUs are designed for high-throughput computing tasks with a fixed architecture, whereas FPGAs are reconfigurable and can create customized data paths to optimize for specific computations which is more performant with the way we resolved this lab.

Please report the FPGA resources (LUT/FF/DSP/BRAM) usages, in terms of resource count and percentage of the total. Which resource has been used most, in terms of percentage?

BRAMs (Block RAMs) are the most utilized resource in terms of percentage, with 36% of BRAMs being used. In contrast, the Look-Up Tables (LUTs) are used at 9%, Flip-Flops (FFs) at 6%, and DSP slices at 15%. Since BRAM is the highest in terms of percentage, it indicates that memory storage is a significant part of this design, likely due to the storage requirements for the weight matrices and intermediate computations in the CNN. Optimizing BRAM usage could be a key focus for further improving the design's efficiency.