Gregory Payton
CS133 w/ Jason Cong
5/22/2025

Lab 4: CNN on FPGA

Starter-Kit	Optimized code	Speedup
5.84 GFlops	267.83 GFlops	45.9x

Parallelization & Optimization Strategies

For the inner kernel loops p & q, the strategy is to fully unroll them and array partition such that every instruction can be done in parallel. For the next innermost loops h & w, the strategy was to do strip mining such that we can fully unroll these new h* and w* loops without utilizing too much hardware.

Evaluation of each Optimization

Pre-Optimization

- Latency: 7.3 billion cycles
- 5.84 GFlops
- 1% DSP utilization (73)

Everything is sequential here. This is very slow and we do not make use of the hardware.

Optimization 1: Loop unrolling p & q convolution loops + array partitioning

- Latency: 3.34 billion cycles
- 12.75 GFlops
- 1% DSP utilization (129)

Performance increased because 5x5 kernel computations are now being done in parallel. However DSP usage is still low. We can unroll a lot more to increase DSP utilization.

Optimization 2: Strip mining h & w loops (by 4,8 respectively) + unrolling +array partitioning

- Latency:159 million cycles
- 267.83 GFlops
- 58% DSP Utilization (4005)

By unrolling more of the loops we were able to increase the number of parallel instructions executed, increasing DSP utilization and decreasing overall latency.

Difference from CUDA (Lab 3)

This lab differs fundamentally from the previous lab because we are programming an FPGA not a GPU. Although we still use C++ to code it is converted by HLS into Verilog code that declares the exact hardware implementation that we want. Rather than write kernel code that is suited for the GPU architecture, we are essentially creating our own architecture on the FPGA that is customized for the CNN code. Thus parallelization is achieved by instruction-level parallelism implemented by the hardware and not thread-level parallelism.

FPGA Resource Usage

The final optimized code has the following hardware usage:

BRAM	DSP	FF	LUT	URAM
1800/4320	4005/6840	444743/788160	256716/394080	25/960
41%	58%	18%	21%	2%

Most of the optimization focused on unrolling and pipelining. This explains why our highest usage is among DSPs.

Bonus: DSP and BRAM Usage (5pts)

Based on the loops, we unrolled about 4*8*5*5 = 800 instructions. There are 6480 DSPs and each instruction takes 5 DSPs so we can have a maximum of 1368 instructions. Therefore we expect 58.48% utilization which matches what we see.

For BRAM we expect to use 1710.6 which is very close to the 1800 used. The calculations are as follows. Input - 50.76K = 2.82 BRAM * 96 splits. Output = 784K = 43.55 BRAM * 32 splits. Weight = 100K = 5.555 BRAM * 25 splits. Summing these gives you 1710.6. This is assuming that each BRAM can only hold one of the arrays.

```
for (int w0 = 0; w0 < 224/8; w0++) {
  #pragma HLS pipeline
  for(int h1 = 0; h1 < 4; h1++){
  #pragma HLS unroll
    for(int w1 = 0; w1 < 8; w1++){
    #pragma HLS unroll
      for (int p = 0; p < 5; p++) {
      #pragma HLS unroll
        for (int q = 0; q < 5; q++) {
        #pragma HLS unroll
          /**
          * Compute the effective output channel index.
          * i0 selects the outer tile and i1 selects the sub
          * within the tile.
          */
          int i = i0 * 16 + i1;
          int h = h0 * 4 + h1;
          int w = w0 * 8 + w1;
          output[i1][h][w] +=
              weight[i1][j][p][q] * input[0][h + p][w + q];
        }
      }
    }
  }
```

Shows the 4*8*5*5 unrolling…