James Bickerstaff ECE 2195 Final Report

V0 of CNN Accelerator

This version of the CNN run on FPGA does not make use of any optimizations of any kind, rather, it sequentially reads in the necessary input and weight data sequentially before doing the computations and writing out the result. This provides poor performance overall, and results in a massive slowdown in comparison to V1, as well as the multicore CPU implementation. Figure 1 below showcases the output from running the generated bitstream on a Xilinx U200 device.

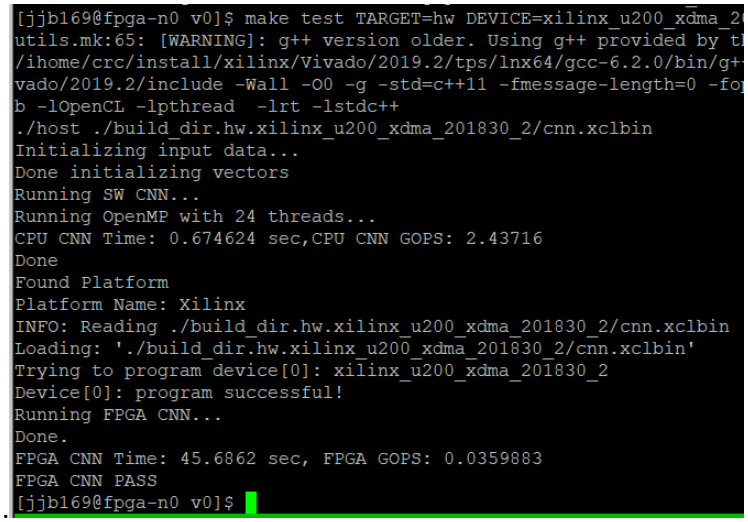

Figure 1. Results from CNN Accelerator V0 on Xilinx U200

Despite computing the correct result, it may be seen that 45.6862 seconds were taken in order to fully complete execution of this CNN operation, producing only 0.0360 GOPS. When compared to the CPU implementation, this is a slowdown of approximately 67x. Different strategies were employed to improve greatly upon this runtime for V1. In terms of an analytical model, this system may be broken down into 3 main components: load, compute, and store. Assuming a memory access time of 110ns (33 cycles), loading data may be modeled as: Clock cycles = (33 * 116 * 116 * 64) + (33 * 4 * 4 * 64 * 64), where the first portion is for loading the input data, and the second is for loading weights. The computation may be modeled as: Clock cycles = (112 * 112 * 64) + (112 * 112 * 64 * 64 * 4 * 4 * 19), where the first component is initializing outputs, and the second is the computations. The ‘19’ at the end of the second factor is an approximation of how many cycles are needed to compute the MAC function in use, and was derived by a generated report as the depth of pipelining the function in V1. Finally, the write back to memory may be modeled as: Clock cycles = (112 * 112 * 64 * 31). The ‘31’ in use for this equation is also derived from the depth of pipelining taken from V1 reports. The values of 112, 116, 64, and 4 used above are the bounds for the various loops used to load, store, and compute data.

The resource utilization for this baseline implementation may be seen within the table below. This design uses very limited resources, under 5% in all categories, with the exception of URAM for storing the data on chip. The requested clock frequency for this design was 300MHz, with a period of 3.332ns. However, this timing constraint could not be met, as a WNS of -2.148ns had been generated. Accounting for this WNS value, the final clock frequency achieved for this design was 182.48MHz.

LUT	FF	DSP	BRAM	URAM
7811 [0.76%]	4866 [0.23%]	7 [0.10%]	65 [3.50%]	407 [42.40%]

Table 1. Resource Utilization of V0

V1 of CNN Accelerator

This version of the CNN accelerator on FPGA makes use of several different optimizations to improve the performance greatly over V0. First, multiple pragmas were added to the underlying data structures used to represent the information within the system. One primary change made to the memory is the partitioning of the 2 arrays used for the input weights and output results. This was done utilizing the complete partitioning method and was specified for the third dimension in particular. This allowed for complete pipelining and unrolling within the primary computational loop. On top of that, 2 port URAM was used for the output array, ensuring that data could be calculated and stored simultaneously.

In addition to the data structure optimizations, the functions for reading data from memory and writing results back to memory were also improved. The process of reading in data from memory (inputs and weights) and storing it within local arrays was accelerated by pipelining each function, ensuring that after the initial latency cost a new piece of data was read in every clock cycle. This was confirmed as working with an initiation interval (II) of 1 by looking further into the reports. Following the completion of the computations, the results are stored back into the host memory in a fashion similar to how the data was initially read. Three nested for loops are utilized due to the dimensionality of this data, and the innermost loop was pipelined with a targeted II of 1. This targeted value had been met and utilized a depth of 31 as well.

While optimizing the memory reads and writes helps improve the runtime, the primary part of the program that requires the most work, and is thus the bottleneck, is the CNN computational loops. In order to initialize the entire output array to 0 before performing computations, this process was both pipelined and unrolled to maximize the number of initialized values every cycle. The II generated for this function was also 1. The last portion to be optimized within the code was the actual computations, which are represented as six nested for loops. To accommodate for complete loop unrolling and pipelining, the 3 loops used to index the output array (h, w, i) were moved from the outermost layers to the innermost portions of the nested loops. This ensured that entire calculations for given indices may be done simultaneously, and after paying the initial latency cost, could be completed every clock

cycle. This II was once again confirmed to be 1, with a depth of 19, by observing the generated reports. Once all optimizations had been implemented, the result produced in Figure 2 below was produced.

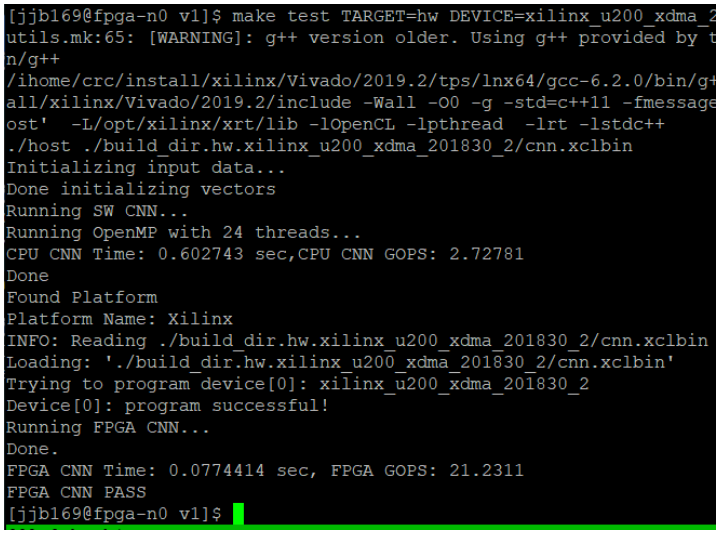

Figure 2. Results from CNN Accelerator V1 on Xilinx U200

The first important part to take note of before observing the runtime results is that the output produced by this FPGA CNN process are correct and match those of the CPU implementation. The results from V1 may be observed in the figure above as 0.0774 seconds and 21.2311 GOPS. Not only is this a 589x speedup over the V0 implementation, but also nearly an 8x improvement over the 24-threaded CPU implementation as well. Following the optimizations made, an analytical model may look similar to the upcoming equations. For data loading: Clock cycles = (33 + (116 * 116 * 64)) + (33 + (4 * 4 * 64 * 64)), the primary difference between this and V0's equation is that the factor of 33 cycles per load is now incurred only once, as the process has been pipelined. Computation may be modeled as:
Clock cycles = (112 * 112 * 1) + (64 * 4 * 4 * 112 * 112 * 1 + 19), the improvements over V0 are that the factor of 64 within the first portion of this equation has become 1, as the innermost loop is now unrolled completely. The second portion of the equation has had a multiply factor of 64 reduced to 1 due to loop unrolling, and the multiplication of 19 has turned to addition due to pipelining, as that latency must only be paid once. The final equation for writing back to memory may look like: Clock cycles = (112 * 112 * 64 + 31). For this equation the 31 cycles needed for a store has been added only once rather than every operation, as the function is now pipelined and the latency cost is paid only on the first store.

The resource utilization for the V1 implementation may be seen below. There is low resource usage across the board, with the exception of URAM, which is nearly at 50% utilization. It should also be noted that the timing constraints could not be met for this design, requesting a clock frequency of 300MHz and a period of roughly 3.332ns. The WNS generated

was -1.859ns, and after taking this into account, the actual clock frequency achieved by V1 of the CNN accelerator was 192.64MHz.

LUT	FF	DSP	BRAM	URAM
26393 [2.57%]	34496 [1.61%]	322 [4.71%]	65 [3.53%]	467 [48.65%]

Table 2. Resource Utilization of V1

Artifacts

Hardware details

The hardware used to run this design was the Xilinx U200 FPGA located on the fpga-n0 node of the Pitt CRC servers.

Software used

The software used for this design was Vitis 2019.2, and all programming was done in the C++ language.

Experiment Workflow

In order to produce the results found above, an iterative approach was taken to optimizing the code and various functions within. A visualization of this process may be seen below:

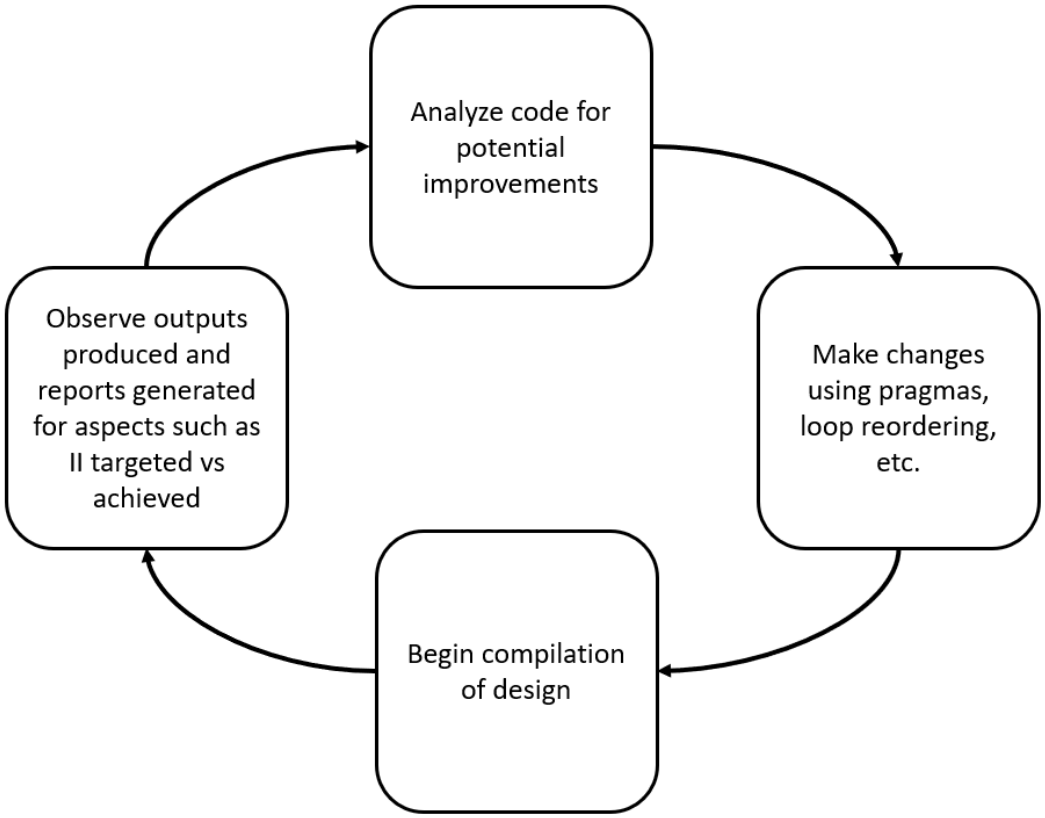

Figure 3. Experimental Workflow Diagram

Evaluation and Expected Results

Overall, there were great improvements made between the V0 and V1 implementations of the CNN accelerator. The performance was as expected after reflecting upon the reports generated during synthesis, as the initiation intervals for all loops that were unrolled and pipelined were able to achieve the targeted value of 1. The only portion of the design that was left somewhat lacking was the targeted frequency. Because the timing constraints could not be met, a negative WNS was generated, and a clock frequency of approximately 192MHz was utilized instead of the targeted 300MHz. Despite this, following implementation of the various optimizations described, the results produced were a great success and improvement over both the V0 and CPU versions of this CNN operation.

Floor plan of V1

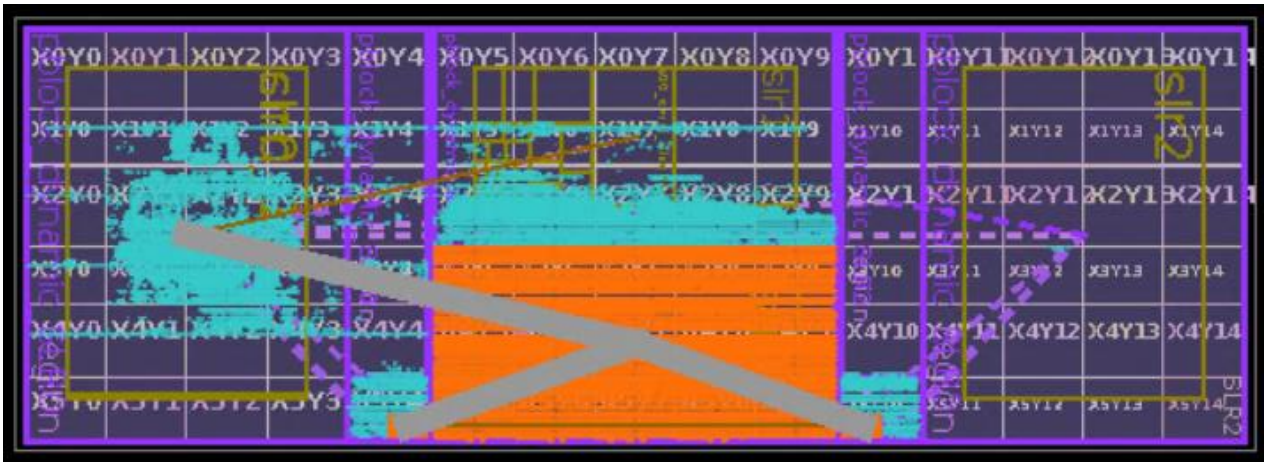

Figure 4. Floorplan of V1 Design

References

All information referenced in the makings of this project was from Dr. Zhou's ECE 2195 Lectures and office hours.