Alec Keehbler, Aksel Torgerson, Max Johnson

Professor Joshua San Miguel

ECE 554

4 October 2021

TPUv1 Design Project

We designed a Tensor Processing Unit (TPU) that can be used for machine learning workloads. The TPU was designed with four modules as well as a control unit.

The first unit we designed was the multiply accumulate unit (MAC). All this unit does is multiply numbers together and add the results. It also sets 2 of its outputs to be the inputs of those two numbers used by other MAC's.

The next unit designed was a systolic array which used the TPUMAC units for computation organized into a 2D array. The inputs to the systolic array are two 2D arrays formed into a parallelogram so that the computations are first done only on the elements in the 0th row and 0th column of the two input 2D arrays. More on that later. The systolic array has the operands read once and then pulse through the array, thus saving energy needed for the delivery of the operands. Every clock cycle, the parallelogram is pushed further into the systolic array, and eventually all values will have had a computation done on them, like a matrix multiply.

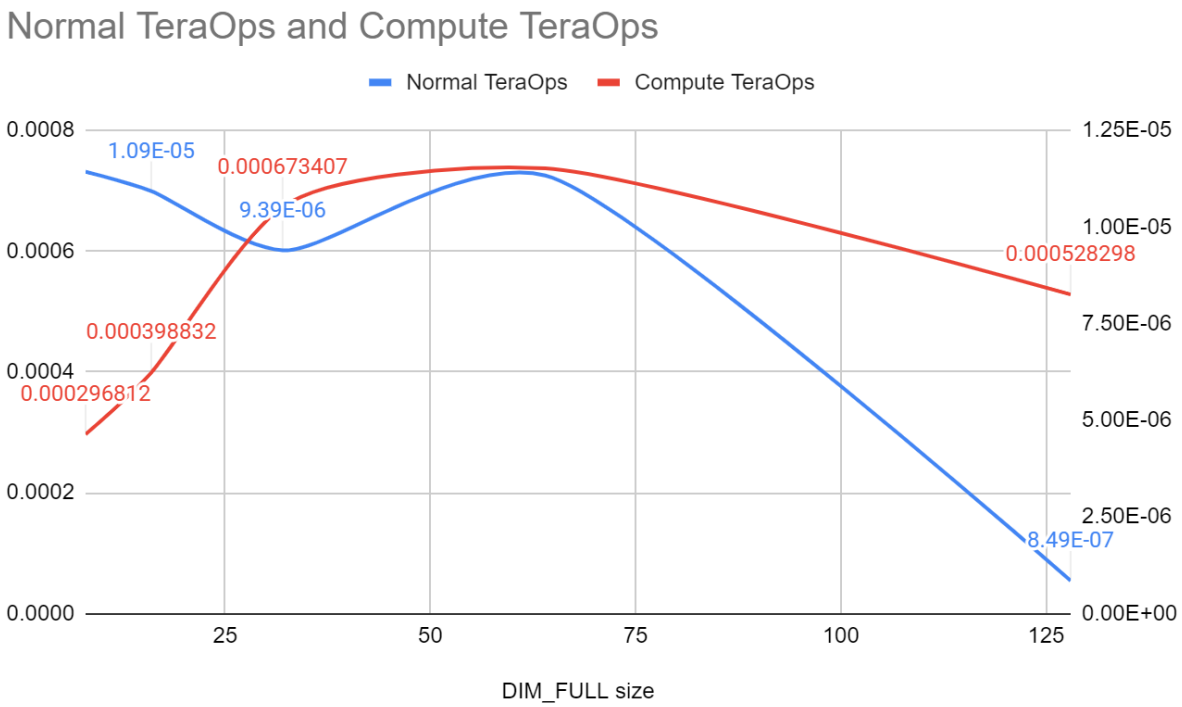
We then made two memory modules for the inputs of the systolic array. These memory modules were made of two types of fifos. The memory module shifting bits in from the "top" used an array of different sized regular fifos. The memory module shifting bits in from the "left" used an array of transpose fifos, which loaded a specified fifo (row) with the input data. The fifos were instantiated in ascending length to create that "parallelogram" input effect. These memory modules would shift out a single vectorized element from a column or row of the respective

operand for the array every clock cycle. The memory modules were loaded based on input addresses from an outer communication module (replicates a PCIe interface).

The TPUv1 module instantiated all of these modules previously talked about to put everything together. With the instantiations, this module acts as a control unit for when to shift the operands and when to start calculation. All of this is what makes up our design for the Tensor Processing Unit.

Our test strategy involved a series of simulations and a few software tests post-synthesis. Each module had its own simulation test where values were rigorously tested against expected functionality along with expected values after resets. The first thing tested was the original fifo, where values were inserted, and after a certain delay the output values were tested against the expected output. The MAC units were then tested rigorously by looping through all possible values it could do computations on. The expected results for the computations were tested against actual output each clock cycle in the double for loop. The next test was for the systolic array. Reset was tested (all values should be zero after reset). A fake A and B input (randomly generated) were used to mimic the parallelogram style input. Another testbench was made for the systolic array but this was in conjunction with the functional memory A and B modules. In this test bench, the functionality of A and B memories were tested. The inputs were in the form of just regular rows/columns of data, and the output was expected to be in the parallelogram format. The outputs to A and B were then the inputs to the systolic array and the computations were tested against expected output. Mem A and B together with the systolic array and control unit were tested post synthesis, using software to input data into A and B and test the output of C.

TPUv1 Results



These results were calculated for each DIM size {8,16,32,64.128}. The Normal TeraOps are the tera ops of the whole process of writing, calculating and reading from the tpu. The Compute TeraOps are the tera ops for only the calculation operations. The compute tera ops were usually a magnitude of 100 bigger than the normal tera ops. These results may not be as accurate due to the limitations of timing libraries in c++, but there is an obvious trend amongst the data. From the compute tera ops data, one can analyze that the peak performance happened when DIM was 64. This is slightly less visible from the normal tera ops, but can still be seen in the line trend where it peaks around 64 for DIM. The drop around 128 also could have been due to overflow of tim values in c++ when computing the timing durations.

The performance of the TPU could drastically be improved by using DMA instead of MMIO. Instead passing in low bandwidth addresses and bytes of data one at a time in order to

write to A and B / read from C, this data could instantly be accessible without constant need for direction from the CPU.