ECE284 Group checkpoint

Systolic Array Accelerator for VGG Net

Zheng Kuang(A59023164), Chen Wang(A59016195),
Qihang Tong(A59019401), Xiaotian Fang(A59026304)

Part 1 Train VGG16 with quantization-aware training

In part 1, we changed the input channel and output channel of features[27] to (8,8), and removed the batch normalization layer after the squeezed convolution.

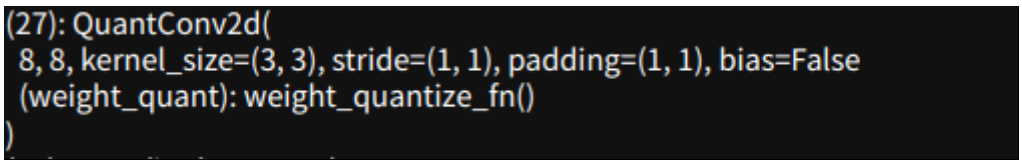


Figure 1. VGG16 Net Overview

Then we train the model get the results:

Accuracy	Psum error
91%	2.89e-7

Table 1. VGG16 Accuracy and Psum Error

Part 2 Complete RTL core design

This part is the complete design of a RTL core, connecting various blocks to finish the convolution task. Central to this design is a 2D array of MAC units, supported by activation and weight input (SRAM) and a multi-banked psum SRAM, for optimized partial sum storage. Unique to this architecture is the integration of an L0 FIFO and an output FIFO, eliminating the need for an IFIFO by directing weight transfer through the L0 FIFO. Additionally, the design incorporates a Special Function Processor, adept at handling accumulation and ReLU operations, ensuring high-speed processing. The corelet.v forms the crux of the design, encompassing all essential blocks like the 2D PE array and the output FIFO, except the SRAMs. This hierarchical structure is for FPGA board for part5. We completed the design requirement and whether there were any compilation errors when connecting all the components.

Part 3

In terms of requested functionality, we do not believe that the operation of the shrinking array requires much external control. Our testbench mainly controls loading data to the kernel, signals the kernel to start computation, and verifies the results, then starts computation and verifies the results.

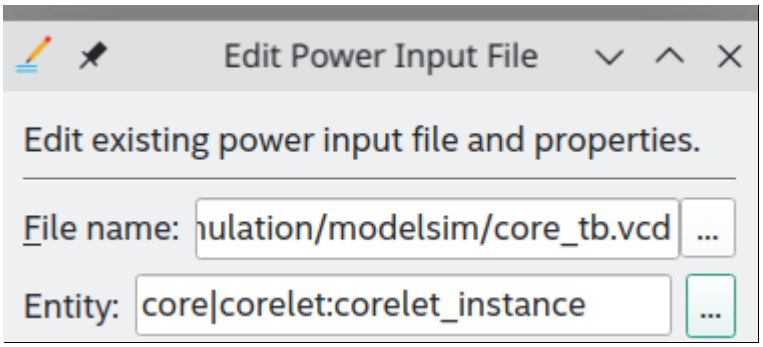
The testbench have fellow steps:

Reading activation file from txt as an array in testbench

1. Writing to SRAM and verify address (Data matched)
2. Reading weight file from txt as an array in testbench
3. Writing to SRAM and verify address (Data matched)
4. Using corelet(controller module) to read the SRAM
5. Sequence calculation Begin
6. Sequence calculation End
7. Red the output file from txt in an array
8. Compare Difference of output and except output in tb, ERROR = 0

Part 4 Mapping on FPGA

In the part 4, We use the modelsim in the Quartus to build the vcd file and using the power analyzer to simulate the power consumption of corelet module in FPGA, in slow mode with 1.2v the core dynamic power is 10.31mw(Chip).



Family	Cyclone IV GX
Device	EP4CGX150DF31I7AD
Power Models	Final
Total Thermal Power Dissipation	207.65 mW
Core Dynamic Thermal Power Dissipation	10.31 mW
Core Static Thermal Power Dissipation	119.37 mW
I/O Thermal Power Dissipation	77.98 mW

Figure 2. Set the corelet

Figure 3. Power simulation result

Frequency	Logic Elements	Register	Gops	Total Power	Dynamic power	Static Power	I/O Power
125.3Mhz	17465	12352	16.03	207.65mW	10.31mW	119.37mW	77.98mW

Table 2. FPGA simulation result

Part 5 Multi-channel design

In this part, we retrained the model to expand the input channel number to 16. To process more input channels, we did some modification to the design. First, the two input channels

separated by 8 are packed in 8 bits in the input activation and weight files in the following fashion:

```
#time0row15[msb-lsb],time0row7[msb-lst],time0row14[msb-lsb],time0row6[msb-lst],...,time0row8[msb-lst],t
#time1row15[msb-lsb],time1row7[msb-lst],time1row14[msb-lsb],time1row6[msb-lst],...,time1row8[msb-lst],t
#.................#
```

```
#col0row15[msb-lsb],col0row7[msb-lst],col0row14[msb-lsb],col0row6[msb-lst],....,col0row8[msb-lst],c
#col1row15[msb-lsb],col1row7[msb-lst],col1row14[msb-lsb],col1row6[msb-lst],....,col1row8[msb-lst],c
#.................#
```

Figure 4. Data Form Overview

Secondly, the size of each block of the input activation/weight sram is increased to 8 bits, and so is the width of fifos of l0. This way, we can read data of two input channels in one l0 read operation.

The mac tile is modified to accommodate 2 weights of input channels separated by 8. In the weight loading stage, the most significant 4 bits of the input are loaded into b_q_1, and the least significant 4 bits are loaded into b_q_0. One new signal “int_acc_cnt” is introduced to control which input set is given to the MAC. When it is 0, the partial sum is the north input, A is the least significant 4 bits of the input and B is b_q_0. When it is 1, the partial sum is the output of the MAC, A is the most significant 4 bits of the input and B is b_q_1. The cnt signal is inverted each cycle. In such a manner, we can accumulate input activation of two input channels in two cycles and send it to the south.

There are also some changes to l0 and mac array. Since the processing time of each PE is 2 cycles for doubled input channel number, we should read one row every 2 cycles as follows:

```
////////////////// versio
//read a fifo every two

rd_en[0]<=rd;
rd_en_block[0]<=rd_en[0];

rd_en[1]<=rd_en_block[0];
rd_en_block[1]<=rd_en[1];

rd_en[2]<=rd_en_block[1];
rd_en_block[2]<=rd_en[2];

rd_en[3]<=rd_en_block[2];
rd_en_block[3]<=rd_en[3];

rd_en[4]<=rd_en_block[3];
rd_en_block[4]<=rd_en[4];

rd_en[5]<=rd_en_block[4];
rd_en_block[5]<=rd_en[5];

rd_en[6]<=rd_en_block[5];
rd_en_block[6]<=rd_en[6];

rd_en[7]<=rd_en_block[6];
rd_en_block[7]<=rd_en[7];
```

Figure 5. Read Instructions

The same is true for the mac array, i.e., the instruction is sent to the next row every two cycles. Also, the valid signal is no longer merely controlled by the inst_e[1] signal, as the output is valid only when the internal accumulation is completed. Therefore, it is asserted when inst_e[1] is true and int_acc_cnt is false.

The size of the ofifo and psum sram is the same as the vanilla version, which means it will first store 36*9 psums in the sram and accumulate later on. In the alpha part this can be optimized to just 36*2 blocks.

Part 6(Alpha)

SRAM optimization

Firstly, we optimized the SRAM, we separate the active and weight SRAM, after that the activation size is 1024 and the weight size is double buffer 256*2 memory, so we saved 512 SRAM.

Dual core implementation

We instantiate two core module and use two multi-bit synchronization modules to achieve sfp_out data crossing between two clock domain. The multi-bit synchronization modules are implemented by double buffering. FIFO may works better and more robust in this situation. Below is a structure figure of the dual core implementation.

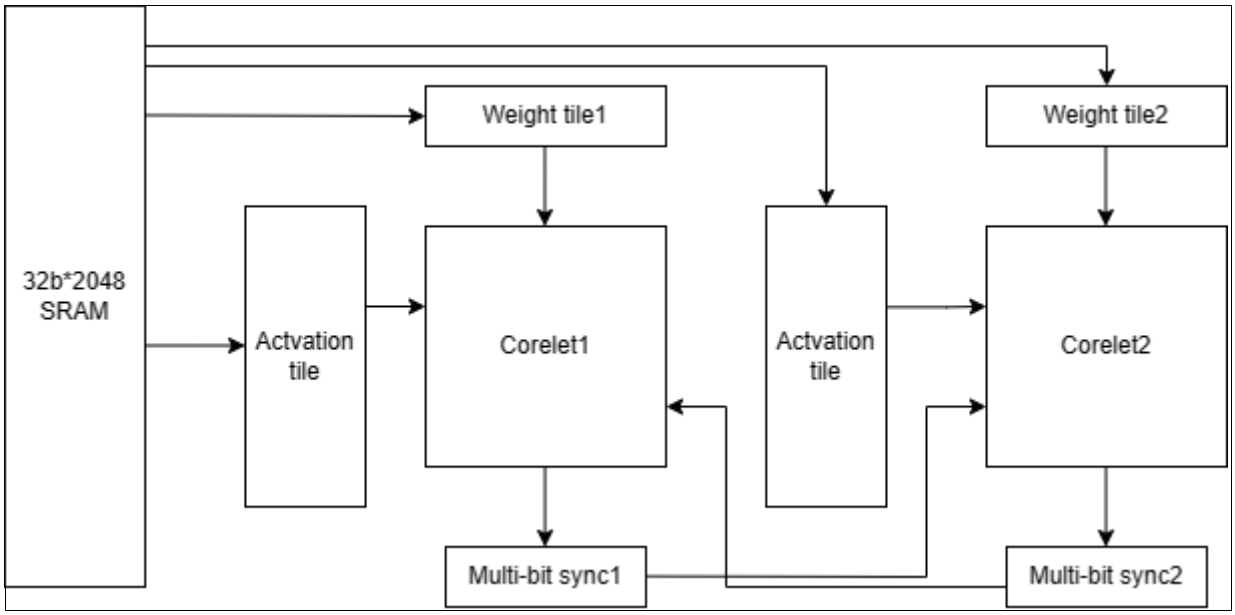


Figure 6. Dual-core 2D Systolic Array Structure

```
multi_bit_sync mb_sync1 (
        .in(sfp_out2),
        .clk(clk1),
        .out(sum_in1)
);
```

Figure 7. Multi-bit Synchronization Interface

Accumulation during execution

To reduce memory consumption, psum can be accumulated right after each kij cycle instead of accumulating until the entire kij-loop is over. This ensures the memory consumption does not scale with the number of input channels.

In this part, another psum sram was added for memory double buffering, and the ofifo, SFP, psum srams are redesigned as illustrated in the figure below.

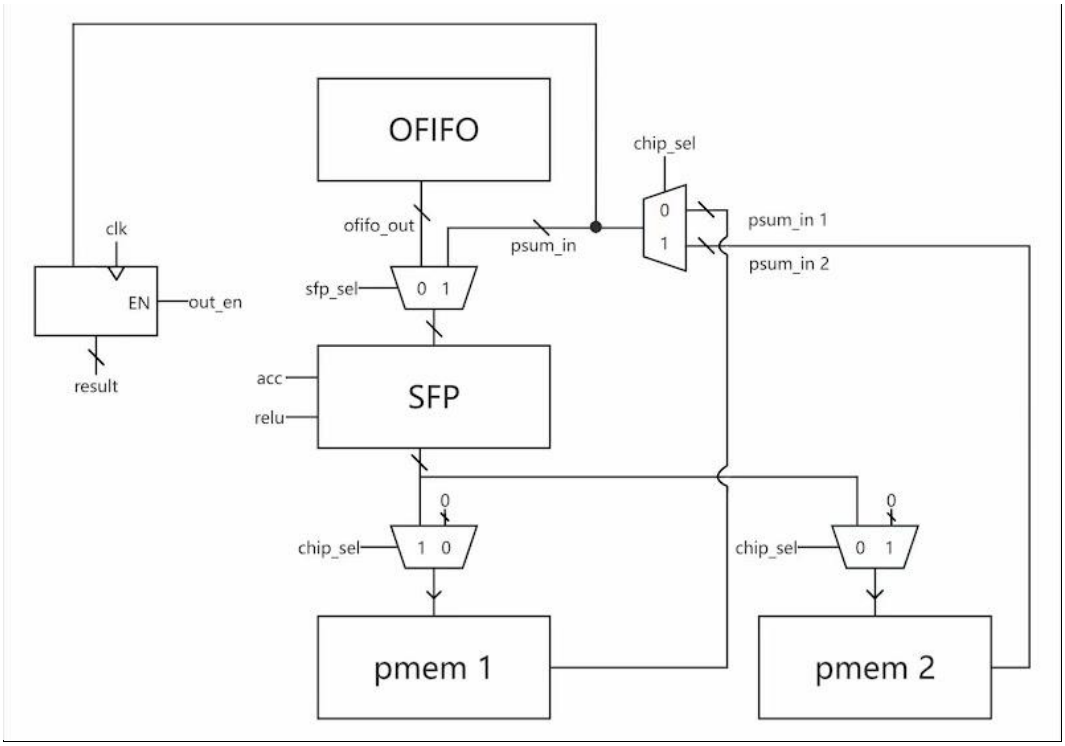


Figure 8. New SFP Design

Sfp_sel signal chooses the input to the SFP between the output from the ofifo or from the psum memory. Chip_sel signal determines which memory to read/write. When it is 0, read pmem1 and write pmem2. When it is 1, read pmem2 and write pmem1. Initially these two signals are 0. We can look at kij=0 as an example. When ofifo outputs one result, it will be sent to SFP. In the next cycle, sfp_sel toggles, so the contents read from pmem1 are sent to SFP. The reading addresses are stored on "accumulation_while_execution_add.txt". Then, the accumulated result is written to pmem2 sequentially. When kij0 is completed, chip_sel toggles, so in the kij1 cycle, contents from pmem2 are used, and the accumulated results are stored in pmem1. The relu signal will be imposed in the last kij cycle. The final result is stored in one of the pmem, as indicated by the chip_sel signal. In this manner, only 36*2 memory blocks are used, and this does not scale with the number of input channels.