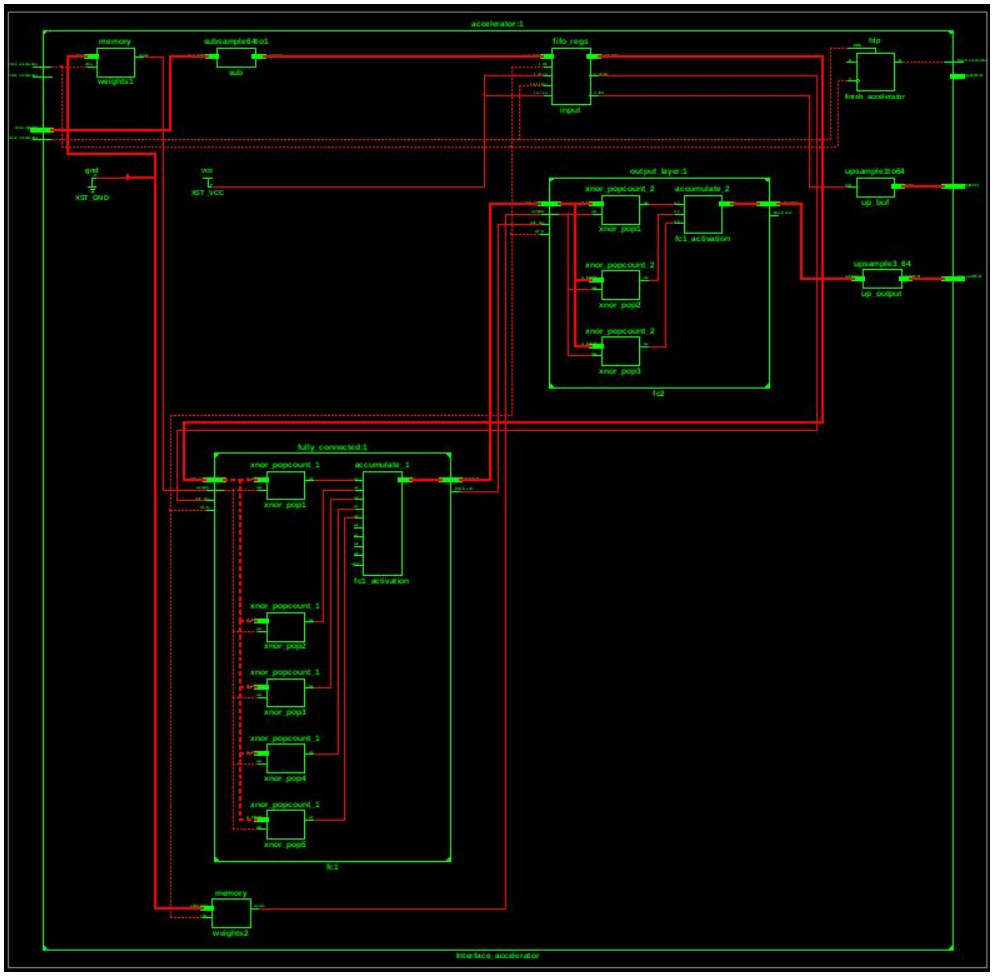
LAB 3 REPORT

Module Implementation

Developing a neural network accelerator



Jannis Wolf - Julien Labarre

16.11.2020

PROCESSOR DESIGN

1. Changes to initial specification

The biggest change to the initial specification is the design as a binary neural network, which means that all activations and weights are either 1 or -1 (implemented as 0). In the following we show the current RTL level of our implementation and describe them more detailed.

Here are some references we used while working. This paper explains the basics of BNNs (Shah). This article shows optimized popcount implementations (VHDL Guru). Here the basics and importance of FIFOs on FPGAs are explained (Nandland).

A. Textual description of the module.

Dataflow in the overall module

We provide a very structured representation of the dataflow of the module in Figure 1. Maybe some components like the subsampling are a bit too detailed as one subcomponent, but it improved the clarity of the signalflow.

Inputs to the *accelerator* are the data of the feature vector, which due to the 64 Bit representation of the TASTE framework has to be subsampled in the module *subsample64to1*, before it is fed into the *fifo*. The *fifo* can indicate to the CPU if it is full over a signal which has to be upsampled to 4 bit in *upsample1to64*. If the first *fully connected layer* sees the non empty signal of the *fifo* it reads the next data into the submodule. In parallel the weight vectors of the layer are read from the *memory* module. The calculated activations flow into the next layer. From the last layer, the activation gets sent back to the CPU through the *upsample3to64* module.

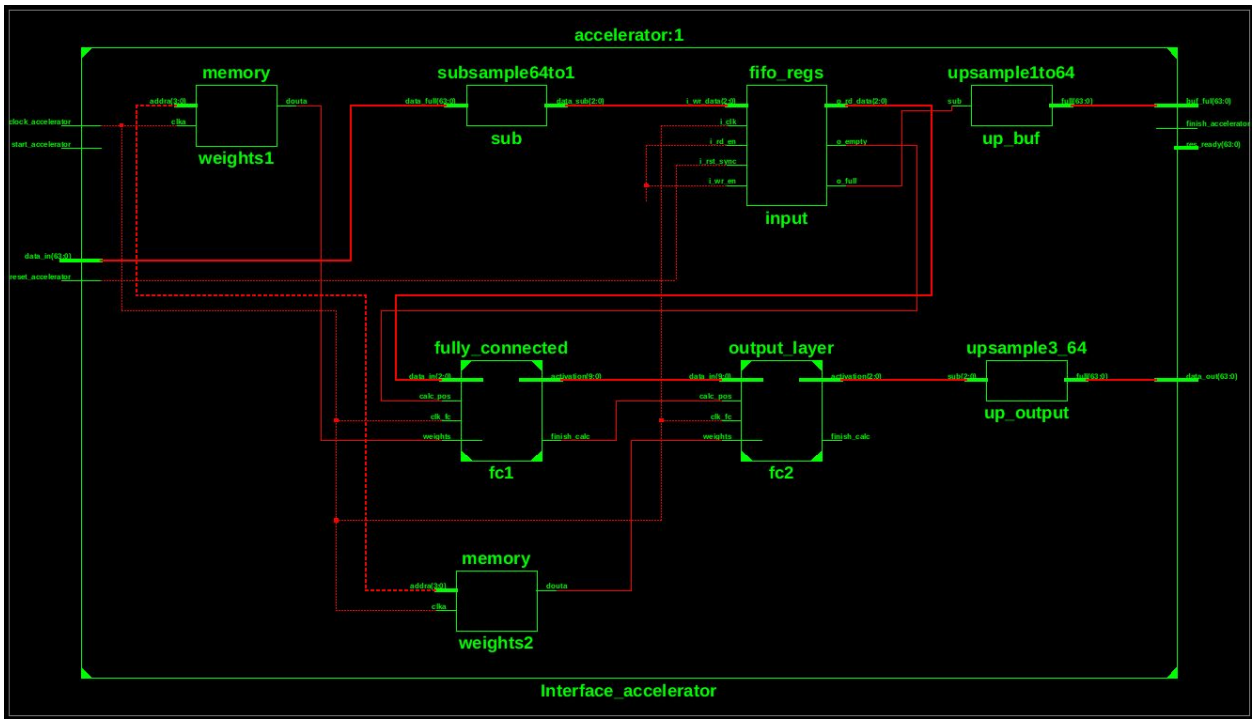


Figure 1: Module interface of the accelerator with internal components.

B. Diagram/s of the module and its internal structure

I. FIFO

As we do not have the hardware yet to test how fast the transfer between the CPU and the accelerator will be and because we wanted to keep the module flexible for bigger input vectors we decided to implement a FIFO to store incoming signals here. At the moment it is register based but with increasing input size BRAM is the better option. The interface is depicted in Figure 2.

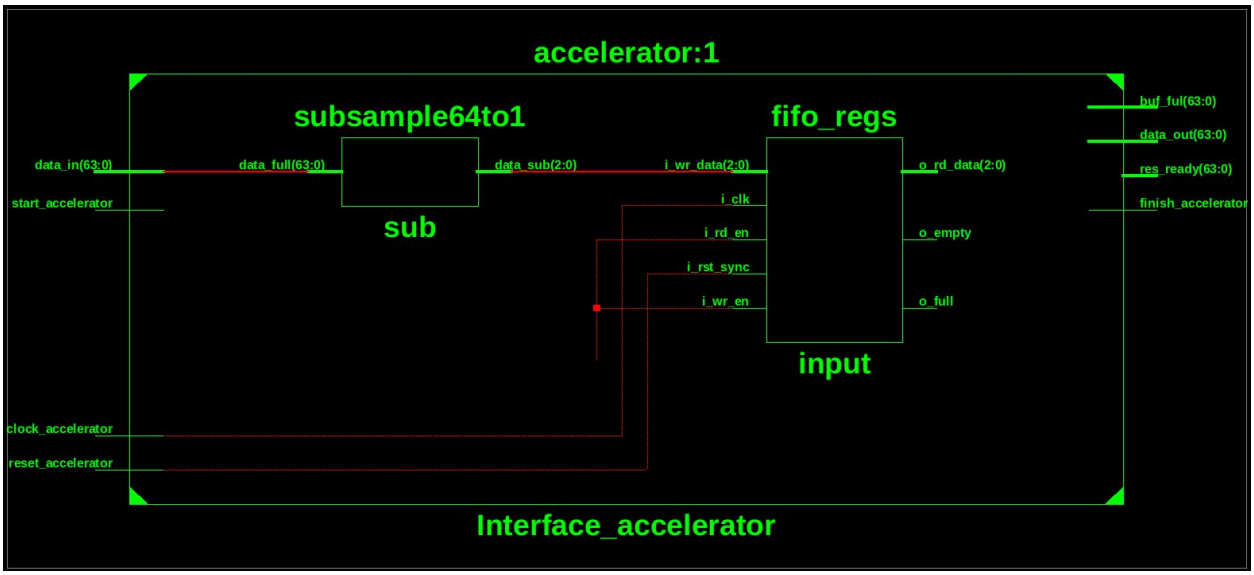


Figure 2: More detailed view on the FIFO module which receives the subsampled input vectors from the CPU.

II. Fully Connected

The fully connected layer receives the data from the FIFO module or from the previous layers and the weights from the memory. The fully connected layer as well as the memory is clocked and connected with an address counter (not depicted in Figure 3). The loading of the weights is a bottleneck as it takes as many cycles as there are neurons in the layer. Figure 3 shows 3 neurons which are basically the *xnor_popcount1* submodules. See (Xu) why the matrix multiplication turns out to be just the popcount of the xnor operation between the input and the weights of the layer. After the calculation an *accumulator* module sends the activations to the next layer. Optimization is possible here with using more BRAM modules for storing and loading the weights.

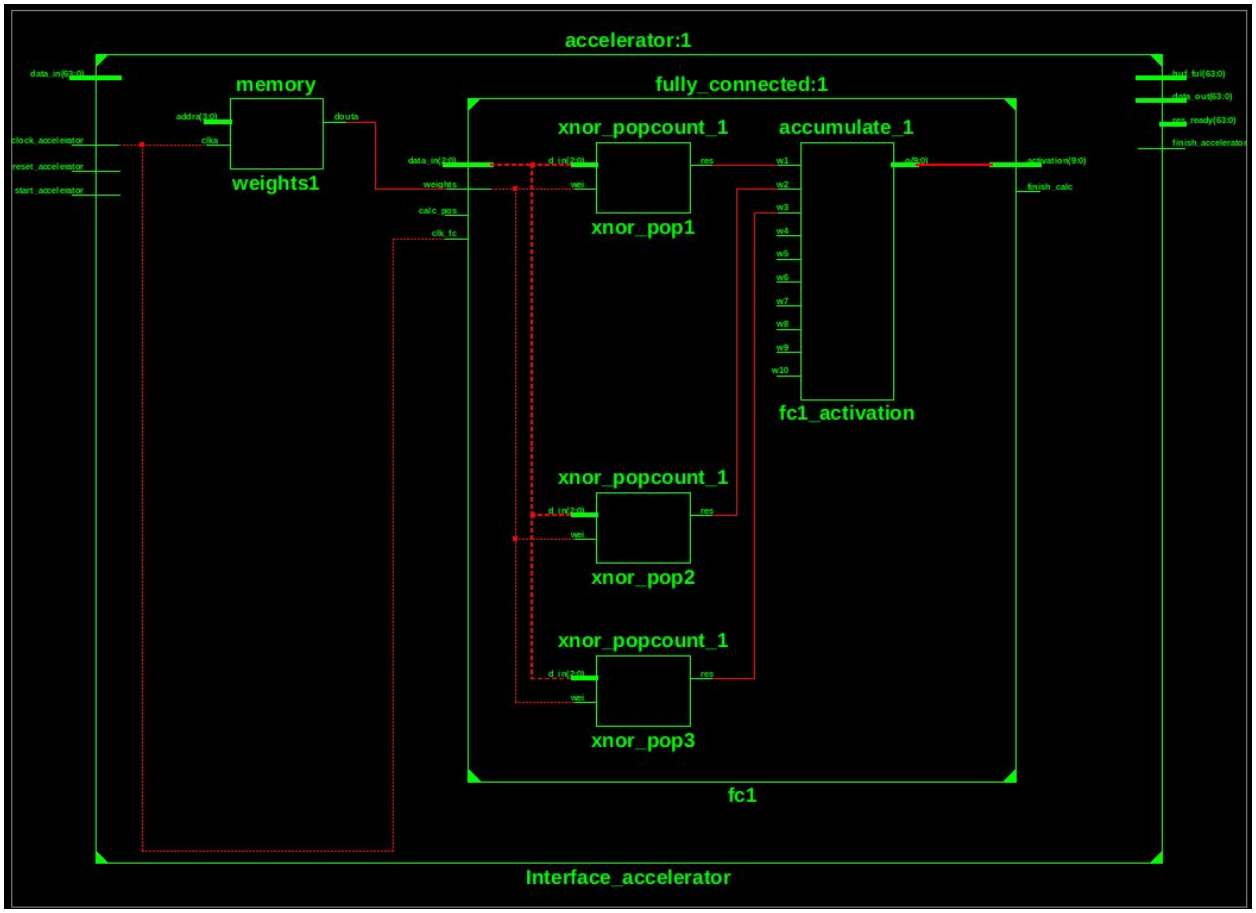


Figure 3: Internals of the fully connected layer and its interface.

C. Definition of inputs and outputs of the module

Input -> 64 Bit integer array

- Taste could just deliver 64 Bit precision, so we subsample the input to 1 Bit precision before feeding it to the FIFO

Output -> integer array

- exact size is determined by the dataset used, but internally the output of the network is a 1 Bit

2. Verification

The output verification does not change significantly. See section 1 for details on input and output, as well as Report 2.

As this is just a quick wrap up of the implementation phase, this report is not too detailed. The next report will present more elaborate results of the project.

3. References

Nandland, Russell. “What is a FIFO in a FPGA?” https://www.nandland.com/articles/what-is-a-fifo-fpga.html. Accessed 16 11 2020.

Shah, Jay. “Binary Neural Networks.” *not published*, 2019, https://github.com/jaygshah/Binary-Neural-Networks/blob/master/about_binary_neural_networks.pdf. Accessed 16 11 2020.

VHDL Guru, vipin. “Count the number of 1's in a Binary number - Circuit design and VHDL implementation.” https://vhdlguru.blogspot.com/2017/10/count-number-of-1s-in-binary-number.html. Accessed 16 11 2020.

Xu, Xianda. “A Computing Kernel for Network Binarization on PyTorch.” *not published*, 2019, https://www.groundai.com/project/a-computing-kernel-for-network-binarization-on-pytorch/1. Accessed 16 11 2020.