IIT JODHPUR

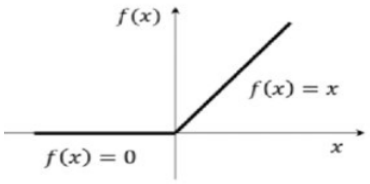
Major Examination: EEL71020 Hardware Design for AI [OPEN-BOOK] (Apr'25)

Guidelines (Total time: 180 minutes, Maximum Marks: 40):

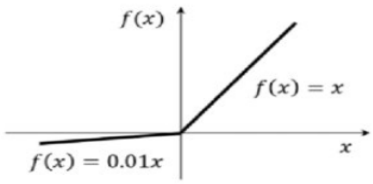
- Please read the question paper very carefully.
- **NO clarification is required in any question.** In case of any doubt, assume whatever you wish to and state that in your answer. Step-wise marks would be awarded wherever applicable.

---

1. Activation functions are frequently used in neural network architectures. Consider the 2 activation functions shown below: For area-constrained and latency restricted model inference, which function is suitable? Why? Design a reconfigurable implementation that can suit both these activation functions? [3 + 3]



ReLU activation function



LeakyReLU activation function

2. Below are 64 weights (in integer represented by 10 bits) ranging from 0 to 1023 that are used in a simple 3-layer artificial neural network (ANN). There is one problem in the implementation of inferences with these weights: we have only 20 bytes storage left in the main memory (where these weights can be stored). To counteract this problem, we can develop an in situ weight-generation process. This process allows on-chip generation of the actual weights from a few weights that would be fetched from the main memory.

```
[0, 16, 32, 49, 65, 81, 97, 113, 130, 146, 162, 178, 194, 210, 227, 243,
 259, 275, 291, 307, 324, 340, 356, 372, 388, 404, 421, 437, 453, 469,
 485, 501, 518, 534, 550, 566, 582, 598, 615, 631, 647, 663, 679, 695,
 712, 728, 744, 760, 776, 792, 809, 825, 841, 857, 873, 889, 906, 922,
 938, 954, 970, 986, 1003, 1019, 1023]
```

Design and describe the on-chip circuitry to realize the above weight clustering-based weight storage in main memory and subsequent usage in the computational engine? [4]

3. Given below is the description of some layers of the widely popular MobileNet V1 deep learning model architecture. Show the steps of the calculation of the below convolution operations and how the size of the successive inputs of intermediate layers get modified from 224x224x3 to 28x28x128? [7]

Type	Stride value	Filter shape	Input size
Standard convolution	Stride = 2	3 × 3 × 3 × 32	224 × 224 × 3
Depthwise separable convolution	Stride = 1	3 × 3 × 32	112 × 112 × 32
Standard convolution	Stride = 1	1 × 1 × 32 × 64	112 × 112 × 32
Depthwise separable convolution	Stride = 2	3 × 3 × 64	112 × 112 × 64
Standard convolution	Stride = 1	1 × 1 × 64 × 128	56 × 56 × 64
Depthwise separable convolution	Stride = 1	3 × 3 × 128	56 × 56 × 128
Standard convolution	Stride = 1	1 × 1 × 128 × 128	56 × 56 × 128
Depthwise separable convolution	Stride = 2	3 × 3 × 128	56 × 56 × 128
Standard convolution	Stride = 1	1 × 1 × 128 × 256	28 × 28 × 128

4. Consider the weights of an intermediate layer in a CNN architecture stored in a filter as 32x32 matrix. Around 75% of these wights are zero resulting in a sparse distribution of the non-zero weights of this model. Measure the benefits in storing this weight matrix in either compressed sparse row (CSR) or compressed sparse column (CSC) format compared to the conventional storage when the weights are stored as FP32 numbers? Considering the above arrangement as baseline, now measure the advantage in storage requirements if the weights are stored as INT8 numbers? [3+2]

5. Develop a parallel processing architecture for the convolution layer operation in a DNN where the input image size is 11x11x3 and the filter size is 5x5x3. Explain the operation of your design with performance analysis assuming clock of 1 GHz? [5+2]

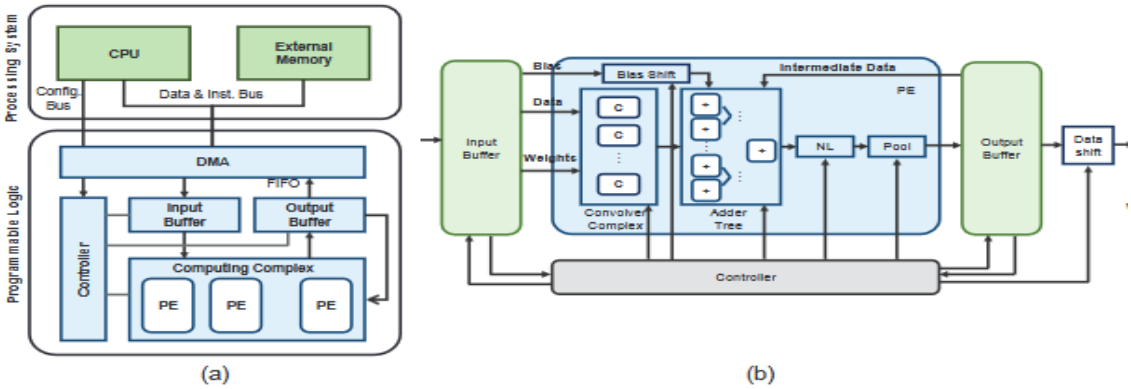
6. Tiling, also known as blocking, is a common optimization technique used in matrix multiplication, especially in the context of General Matrix Multiply (GEMM) in numerical linear algebra. The idea behind tiling is to improve data locality by dividing matrices into smaller sub-matrices or "tiles", and then performing the multiplication operation on these smaller chunks. Design an accelerator for Sparse Matrix-Matrix multiplication (SpMM) that utilizes the below function: [5]

```
function Tiled_Inner(A, B, C, M, N, K, blockSize):
    for m from 0 to M step blockSize:
        for n from 0 to N step blockSize:
            for k from 0 to K step blockSize:

                // Calculate upper bounds for each block
                mUpper = m + blockSize
                nUpper = n + blockSize
                kUpper = k + blockSize

                for i from m to mUpper:
                    for j from n to nUpper:
                        temp = 0
                        for l from k to kUpper:
                            temp += A[i][l] * B[l][j]
                        C[i][j] += temp
end function
```

7. Embedded FPGA boards employ integration of an off-shelf processor (CPU) with the reconfigurable fabric/logic (also referred to as programmable logic, PL). These boards can be utilized to implement neural network model implementations for acceleration objectives.



In the above figure, (a) represents the overall architecture and (b) the processing element in the neural network acceleration architecture. The details are shown as below: (C represents convolution block)

- Adder Tree (AD) sums all the results from convolvers. It can add the intermediate data from Output Buffer or bias data from Input Buffer if needed.
- Non-Linearity (NL) module applies non-linear activation function to the input data stream.
- Max-Pooling module utilizes the line buffers to apply the specific 2×2 window to the input data stream, and outputs the maximum among them.
- Bias Shift module and Data Shift module are designed to support dynamic quantization. Input bias will be shifted by Bias Shift according to the layer's quantization result. For a 16-bit implementation, the bias is extended to 32-bit to be added with convolution result.

Let's assume that the convolver design is same as that of what you have designed in Question no. 5, explain how the overall architecture (Fig. (b)) offers enhanced performance over the conventional accelerator design? (Hint: you may provide the design of adder tree, NL block, pooling block etc.) [6]