Tensor ALU

(Vector generation logic + Vector ALU)

Assignee: Sadhana S

Co-Assignee: Mohan

Description:

Vector generation logic loads input data from the output buffer using input base address and input memory stride and writes the output to the output buffer using output base address and output memory stride. The number of such vector operations and the number of elements in vector depends on the loop extent field in ALU instruction

In Vector ALU,

Operands – Vector X, Vector Y

Output – Vector Z

Operations supported:

1. Min
2. Max
3. Add
4. Sub
5. Shift

Size of Vectors can be equal to the number of systolic columns (Just an arbitrary choice can be modified based on performance)

Tensor ALU instruction Fields:

1. Opcode(2)
2. Dependency flags
3. ALU specific Opcode(4)
4. Base Address of Vector 1(32 bit absolute address or Offset of Output buffer alone can be given)
5. Base Address of Vector 2(32 bit absolute address or Offset of Output buffer alone can be given)
6. Input memory stride(4)
7. Is Immediate available(1)
8. Immediate value(8)
9. Is continuous operation(1)
10. Base address of vector output(32 bit absolute address or Offset of Output buffer alone can be given)
11. Output memory stride(4)
12. Loop extent 0 - Number of elements in the vector
13. Loop extent 1 - Number of times the Vector operations are performed

14. Matrix dimensions - Input dimension - N*C*H*W and operations specific dimension such as Max pool dimension and stride for Max Pooling.

Concerns :

1. Size of ALU instruction is more

Inputs to the module:

1. ALU instruction with all the necessary fields

Outputs from the module:

1. ALU complete signal

Pseudo-code:

1. ALU module
    a. Receives the ALU specific opcode, one vector element from Vector X and corresponding vector element from Y and gives an element of Vector Z
2. Vector of ALU modules
    a. Instantiates ‘n’ ALU modules to operate on ‘n’ vector elements
    b. Gives appropriate inputs to the Vector of ALUs and receives ‘n’ outputs.
3. Vector generation logic
    a. Get data from output buffer for vector operation
    b. Store the output vector to output buffer
    c. Calculate address based on the op-code for next vector operation and repeats for ‘loop count 1’ times

Doubts:

1. With base operations as Add, Sub, Max, Min and Shift Right, Max pooling and ReLU alone is supported. Should there be support for Average Pooling, Normalisation?
2. Is it possible to view the Uop space through compiler generated instructions?

Milestones:

1. Coding ALU module (8/4/2020 - 10/4/2020)
2. Coding Vector ALU module (11/4/2020 - 12/4/2020)
3. Little more prelim design work on Vector generation logic (9/4/2020 - 13/4/2020)
4. Coding the Vector generation logic(14/4/2020 - 20/4/2020)