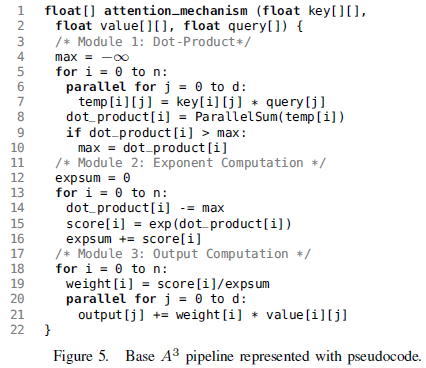
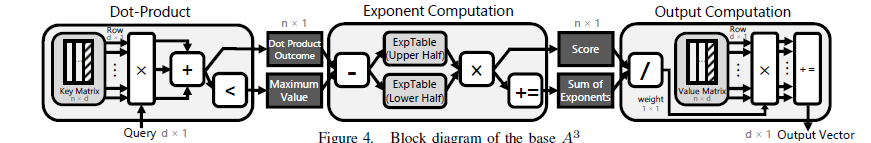
Neural networks are one of the most popular techniques to perform complex AI tasks in various domains including computer vision and natural language processing. With large amount of data, neural networks can solve a wide range of AI challenges and can often surpass human performance in many domains. However, these advantages come at a high computational cost involving tens of billions of operations. In order to minimize the energy cost of such operations, and maximize the throughput of neural network processing, many FPGA and ASIC based accelerators have been proposed. Most of these works have targeted the acceleration of CNNs and RNNs and have shown significant performance improvements. However, these accelerators for not provide full support for the emerging neural network primitive such as attention mechanism in the Transformer models.

An attention mechanism is usually implemented as dense matrix operations and softmax operations. Matrix multiplication operation computes the similarity across search targets and thus the computational complexity of this operation is proportional to the number of search targets. A3 is a hardware accelerator for attention mechanism. It employs pipelined design with customized datatpath for high throughput and energy efficiency.

**Pipeline Design**

A3 takes three inputs – a key matrix (x), a value matrix (x), and a query vector ()- to compute dimension output vector.





**Module 1 – Dot Product Computation**

* The first module of the pipeline computes the inner-product between a single row of the key matrix and a query vector.
* The hardware is consists of multipliers and a -way adder tree for a sum reduction operation.
* For each cycle, a row of the key matrix is loaded (in sequential order) and each of its vector element is multiplied by the corresponding element of the query vector using the array of multipliers.
* After multiplication, the set of values are passed to adder tree for parallel sum reduction.
* The result is stored in the corresponding register in the dot-product outcome register file.
* This module also finds the maximum value among all the elements of the dot-product array.

This primary arithmetic units in this module are the multipliers and adders. The conventional arithmetic units can be replaced by the online arithmetic units. The digit-level pipelining property of online arithmetic can be exploited for an overlapped multiplication followed by addition, resulting in reduced number of computation cycles, consequently reducing the latency.

**Module 2 – Exponent Computation**

* This module computes the exponent of each dot-product value computed by Module-1.
* Exponent function is implemented using a look-up table.
* The module first starts with subtracting the maximum value of the input vector from the dot-product value being processed.
* With this subtraction all elements of the input vector are normalized between 0 and 1.
* To reduce the size of look-up table, the exponent function is decomposed into multiplication of two exponent operations.
* With this transformation, two smaller look-up tables can be utilized with a multiplication, instead of a one large look-up table.
* After exponent computation, the value is accumulated for later use as softmax denominator.

The softmax computation can be performed using a look-up table as in A3. The look-up table strategy can be improved if online arithmetic is employed since the most significant digits are obtained first. A threshold can be set which would primary arithmetic units in this module are the multipliers and adders. The conventional arithmetic units can be replaced by the online arithmetic units. The digit-level pipelining property of online arithmetic can be exploited for an overlapped multiplication followed by addition, resulting in reduced number of computation cycles, consequently reducing the latency.

**Module 3 – Output Computation**

* This module computes the output of the attention operation.
* Every cycle, an element of the score vector is divided by the sum of all score values for normalization.
* Then this value (weight) serves as a scaling factor for the corresponding row vector in the value matrix.
* Each element of a row vector is multiplied by this value.
* The result of this computation is accumulated in the output register and the process is repeated n times.

This module performs similar computations as in Module 1, suggesting the use of similar type of hardware. The A3 architecture uses the three modules in a pipelined manner in order to improve the throughput. The online multiplier with pipelined configuration can be utilized to further enhance the throughput.