CLAIMS

1. A convolution acceleration method, comprising:

reading sequentially, through a First Input First Output (FIFO) port, data to be processed of a preset size in a task to be processed from an off-chip memory of an accelerator, and storing the data to be processed into at least one input buffer of a first buffer in an on-chip memory of the accelerator, wherein the on-chip memory comprises the first buffer, a second buffer and a third buffer, the first buffer comprises at least one input buffer, and at least one set of convolution kernel data is buffered in the second buffer;

in response to the data to be processed being stored into an input buffer, reading input window data to be processed from the at least one input buffer sequentially, performing convolution on the input window data to be processed by using the convolution kernel data to obtain output window data and storing the output window data in the third buffer; and

storing, through the FIFO port, the output window data in the third buffer into the off-chip memory sequentially.

2. The method according to claim 1, wherein the task to be processed comprises an input image of at least one input channel, or an input feature map of the at least one input channel; and

each set of convolution kernel data corresponds to one output channel, each set of convolution kernel data comprises at least one convolution kernel data, and each convolution kernel data corresponds to one input channel.

3. The method according to claim 1 or 2, wherein the third buffer comprises at least one output buffer;

the obtaining output window data and storing the output window data in the third buffer comprises: obtaining the output window data and storing the output window data in the at least one output buffer sequentially; and

the storing, through the FIFO port, the output window data in the third buffer into the off-chip memory sequentially comprises: in response to output window data being stored in a first output buffer, reading output window data in the at least one output buffer sequentially, and storing the read output window data into the off-chip

memory through the FIFO port.

4. The method according to claim 3, wherein the third buffer comprises two output buffers; and

during processing the task to be processed, storage of output window data and reading of output window data are performed on one of the two output buffers alternately, and reading of output window data and storage of output window data are performed on the other of the two output buffers alternately.

5. The method according to any one of claims 1 to 4, wherein the first buffer comprises two input buffers; and

during processing the task to be processed, storage of the data to be processed and reading of the input window data to be processed are performed on one of the two input buffers alternately, and reading of the input window data to be processed and storage of the data to be processed are performed on the other of the two input buffers alternately.

6. The method according to any one of claims 1 to 4, wherein a size of a convolution kernel corresponding to the convolution kernel data is r×r, a size of an input window corresponding to the input window data is n×n, and a sliding step of the input window corresponding to the input window data and a size of an output window corresponding to the input window data are m, wherein values of m, n, and r are respectively integers greater than 0 and n=m+r-1;

the input buffer is a line buffer; the first buffer is a cyclic buffer formed by n+m line buffers, and each of the line buffers is used for storing sequentially data in the same row in all channels of the task to be processed; and

the reading data to be processed of a preset size in a task to be processed from an off-chip memory of an accelerator and storing sequentially the data to be processed into at least one input buffer of a first buffer in an on-chip memory of the accelerator comprises:

reading the first n rows of data in all channels of the task to be processed from the off-chip memory for the first time, and storing correspondingly each row of data in the first n rows of data into the first n line buffers determined according to a predetermined cyclic direction in the first buffer; and

then reading m adjacent subsequent rows of data in all channels of the task to

be processed from the off-chip memory each time, and storing correspondingly the m rows of data read this time into subsequent m adjacent line buffers determined according to the predetermined cyclic direction in the first buffer;

or,

the reading data to be processed of a preset size in a task to be processed from an off-chip memory of an accelerator and storing sequentially the data to be processed into at least one input buffer of a first buffer in an on-chip memory of the accelerator comprises:

reading n rows of data in all channels of the task to be processed from the off-chip memory each time;

for the first n rows of data read from the task to be processed for the first time, storing correspondingly the first n rows of data into the first n line buffers determined according to a predetermined cyclic direction in the first buffer; and for n rows of data that are read not for the first time, taking n corresponding line buffers as target line buffers for the n rows of data read this time, wherein the n corresponding line buffers are obtained by sliding the first n line buffers, which store n rows of data read last time, by m rows according to the predetermined cyclic direction in the first buffer, and storing the n rows of data read this time into the target line buffers correspondingly.

7. The method according to claim 6, wherein the reading input window data to be processed from the at least one input buffer sequentially and performing convolution on the input window data to be processed by using the convolution kernel data comprises:

for the task to be processed, when the input window data to be processed is read for the first time, selecting the first n line buffers as a current computing buffer, reading input window data with a size of n×n from the current computing buffer sequentially according to the sliding step m, and performing convolution on the input window data to be processed by using the convolution kernel data until all data to be processed in the current computing buffer is read; and

then selecting n line buffers as a new computing buffer via sliding from the previous computing buffer according to the sliding step m and the predetermined cyclic direction each time, and reading input window data with a size of n×n from the new computing buffer sequentially according to the sliding step m, and performing convolution on the input window data to be processed by using the convolution kernel

data until all data to be processed in the new computing buffer is read.

8. The method according to claim 6 or 7, wherein the output buffer comprises m line buffers;

the obtaining the output window data and storing the output window data in the at least one output buffer sequentially comprises:

selecting one of the output buffers as a current output buffer, and storing sequentially output window data, obtained by performing convolution on each input window data, into the m line buffers of the current output buffer; and in response to storage of output window data, obtained by performing convolution on all input window data in one computing buffer, being completed, selecting another output buffer as the current output buffer, and performing the operation of storing sequentially output window data, obtained by performing convolution on each input window data, in the m line buffers of the current output buffer; and

the storing the output window data in the third buffer into the off-chip memory sequentially comprises:

in response to storage of output window data, obtained by performing convolution on all input window data in the current computing buffer, being completed, storing the output window data in the current output buffer into the off-chip memory.

9. The method according to any one of claims 1 to 4, wherein a size of a convolution kernel corresponding to the convolution kernel data is r×r, a size of an input window corresponding to the input window data is n×n, and a sliding step of the input window corresponding to the input window data is m, wherein the values of m, n, and r are respectively integers greater than 0 and n=m+r-1;

the input buffer is a column buffer; the first buffer is a cyclic buffer formed by n+m column buffers, and each column buffer is used for sequentially storing data in the same column in all channels of the task to be processed; and

the reading data to be processed of a preset size in a task to be processed from an off-chip memory of an accelerator and storing sequentially the data to be processed into at least one input buffer of a first buffer in an on-chip memory of the accelerator comprises:

reading the first n columns of data in all channels of the task to be processed

from the off-chip memory for the first time, and storing correspondingly each column of data in the first n columns of data into the first n column buffers determined according to a predetermined cyclic direction in the first buffer; and

then reading m adjacent subsequent columns of data in all channels of the task to be processed from the off-chip memory each time, and storing correspondingly the m columns of data read this time into subsequent m adjacent column buffers determined according to the predetermined cyclic direction in the first buffer;

or,

the reading data to be processed of a preset size in a task to be processed from an off-chip memory of an accelerator and storing sequentially the data to be processed into at least one input buffer of a first buffer in an on-chip memory of the accelerator comprises:

reading n columns of data in all channels of the task to be processed from the off-chip memory each time;

for the first n columns of data read from the task to be processed for the first time, storing correspondingly the first n columns of data into the first n column buffers determined according to a predetermined cyclic direction in the first buffer; and for n columns of data that are read not for the first time, taking n corresponding column buffers as target column buffers for the n columns of data read this time, wherein the n corresponding column buffers are obtained by sliding the first n column buffers, which store n columns of data read last time, by m columns according to the predetermined cyclic direction in the first buffer and storing the n columns of data read this time into the target column buffers correspondingly.

10. The method according to claim 9, wherein the reading input window data to be processed from the at least one input buffer sequentially and performing convolution on the input window data to be processed by using the convolution kernel data comprises:

for the task to be processed, when the input window data to be processed is read for the first time, selecting the first n column buffers as a current computing buffer, reading input window data with a size of n×n from the current computing buffer sequentially according to the sliding step m, and performing convolution on the input window data to be processed by using the convolution kernel data until all data

to be processed in the current computing buffer is read; and

then selecting n column buffers as a new computing buffer via sliding from the previous computing buffer according to the sliding step m and the predetermined cyclic direction each time, and reading input window data with a size of n×n from the new computing buffer sequentially according to the sliding step m, and performing convolution on the input window data to be processed by using the convolution kernel data until all data to be processed in the new computing buffer is read.

11. The method according to claim 9 or 10, wherein the output buffer comprises m column buffers;

obtaining the output window data and storing the output window data in the at least one output buffer sequentially comprises:

selecting one of the output buffers as a current output buffer, and storing sequentially output window data, obtained by performing convolution on each piece of input window data, in the m column buffers of the current output buffer; and in response to storage of output window data, obtained by performing convolution on all input window data in one computing buffer, being completed, selecting another output buffer as a current output buffer, and performing the operation of storing sequentially output window data, obtained by performing convolution on each piece of input window data, in the m column buffers of the current output buffer; and

the storing the output window data in the third buffer into the off-chip memory sequentially comprises:

in response to storage of output window data, obtained by performing convolution on all input window data in the current computing buffer, being completed, storing the output window data in the current output buffer into the off-chip memory.

12. The method according to any one of claims 1 to 11, wherein the at least one set of convolution kernel data comprises at least one set of converted convolution kernels, wherein any set of converted convolution kernels is obtained by converting convolution kernels corresponding to one output channel through a corresponding first constant matrix; and

the performing convolution on the input window data to be processed by using the convolution kernel data to obtain output window data comprises:

performing the following computing processing operations for any input channel respectively: converting the input window data to be processed through a corresponding second constant matrix to obtain converted input window data; performing a matrix dot product operation on the converted input window data of the any input channel and the converted convolution kernels to obtain a matrix subjected to dot product; and converting the matrix subjected to dot product through a corresponding third constant matrix to obtain a convolution result; and

in response to obtaining convolution results of all input channels, accumulating corresponding points in the convolution results of all input channels to obtain output window data of one output channel.

13. The method according to claim 12, further comprising:

converting respectively convolution kernels corresponding to the at least one output channel through the corresponding first constant matrix in advance to obtain the at least one set of converted convolution kernels, and storing the converted convolution kernels in the second buffer; or

converting respectively convolution kernels corresponding to the at least one output channel through the corresponding first constant matrix to obtain the at least one set of converted convolution kernels.

14. The method according to claim 12 or 13, wherein the performing the following computing processing operations for any input channel respectively comprises: simultaneously performing the computing processing operations for Pm output channels and Pn input channels, wherein Pm and Pn are respectively integers greater than 0; and

the in response to obtaining convolution results of all input channels, accumulating corresponding points in the convolution results of all input channels to obtain output window data of one output channel comprises: for the Pm output channels, in response to obtaining convolution results of all input channels, accumulating respectively corresponding points in the convolution results of all input channels for one output channel to obtain output window data of the Pm output channels.

15. The method according to claim 14, wherein the value of Pm is 1, and the value of Pn is an integer greater than 1; and

the simultaneously performing the computing processing operations for Pm output channels and Pn input channels comprises: sequentially performing the computing processing operations in parallel for one output channel and all input channels, or one output channel and some input channels in all input channels.

16. The method according to claim 15, wherein the value of Pm is the number of all output channels, and the value of Pn is 1;

the simultaneously performing the computing processing operations for Pm output channels and Pn input channels comprises: for all output channels simultaneously, performing sequentially the computing processing operations for one input channel; and

the in response to obtaining convolution results of all input channels, accumulating corresponding points in the convolution results of all input channels to obtain output window data of one output channel comprises: for all output channels, in response to obtaining the convolution results of all input channels, separately accumulating corresponding points in the convolution results of all input channels for one output channel to obtain output window data of all output channels.

17. The method according to any one of claims 14 to 16, further comprising:

obtaining, according to a parameter optimization model, optimal values of parameters in convolution based on a transmission bandwidth of the FIFO port and a resource condition of the accelerator, wherein the parameters in convolution comprise: a size of the input window, a degree of parallelism of the output channel, and a degree of parallelism of the input channel, and the optimal values of the parameters comprise: an optimal value n of the size of the input window, an optimal value Pm of the degree of parallelism of the output channel, and an optimal value Pn of the degree of parallelism of the input channel; and

the simultaneously performing the computing processing operations for Pm output channels and Pn input channels comprises:

selecting Pm output channels and Pn input channels based on the values of Pm and Pn in the optimal values of the parameters, and performing the operation of simultaneously performing the computing processing operations for Pm output channels and Pn input channels.

18. The method according to any one of claims 1 to 17, further comprising:

determining whether the size k×k of a convolution kernel of the current task to be processed is consistent with the size r×r of the convolution kernel corresponding to the accelerator; and

responsive to determining that the size k×k of the convolution kernel of the current task to be processed is not consistent with the size r×r of the convolution kernel corresponding to the accelerator, padding the convolution kernel of the current task to be processed with a columns of 0 on the right and b rows of 0 at the bottom, so that the convolution kernel of the current task to be processed is converted into a new convolution kernel with a size of tr×tr, wherein values of k, t, a, and b are respectively integers greater than 0.

19. A convolution accelerator, comprising:

a data reading module, configured to read sequentially, through a First Input First Output (FIFO) port, data to be processed of a preset size in a task to be processed from an off-chip memory of the accelerator, and store the data to be processed into at least one input buffer of a first buffer;

the first buffer, comprising at least one input buffer and configured to buffer the data to be processed;

a second buffer, configured to buffer at least one set of convolution kernel data;

a computing module, configured to read input window data to be processed from the at least one input buffer sequentially when data to be processed is stored into an input buffer, perform convolution on the input window data to be processed by using the convolution kernel data to obtain output window data and store the output window data in a third buffer;

a third buffer, comprising at least one output buffer and configured to buffer the output window data; and

a data output module, configured to store the output window data in the third buffer into the off-chip memory sequentially through the FIFO port.

20. A computer storage medium, for storing computer-readable instructions, wherein the instructions, when being executed, implements the operations according to any one of claims 1 to 18.