NNgen

Release 1.0.0

Shinya Takamaeda-Yamazaki

Feb 21, 2020

CONTENTS:

CHAPTER

ONE

OPERATOR

class nngen.operator.basic.**add**(*x*, *y*, *dtype=None*, *name=None*, *par=1*)

Returns $x + y$ element-wise.

Parameters

- **x** – A tensor.
- **y** – A tensor.
- **dtype** – Output data type (optional).
- **name** – A name for the operation (optional).
- **par** – The number of parallel operations (optional).

class nngen.operator.basic.**sub**(*x*, *y*, *dtype=None*, *name=None*, *par=1*)

Returns $x - y$ element-wise.

Parameters

- **x** – A tensor.
- **y** – A tensor.
- **dtype** – Output data type (optional).
- **name** – A name for the operation (optional).
- **par** – The number of parallel operations (optional).

class nngen.operator.basic.**add_n**(*arg*, *dtype=None*, *name=None*, *par=1*)

Adds all input tensors element-wise.

Parameters

- **arg** – A list of Tensor objects.
- **dtype** – Output data type (optional).
- **name** – A name for the operation (optional).
- **par** – The number of parallel operations (optional).

class nngen.operator.basic.**lshift**(*x*, *y*, *dtype=None*, *name=None*, *par=1*)

Elementwise computes the bitwise left-shift of x and y.

Parameters

- **x** – A tensor.
- **y** – A tensor.

- **dtype** – Output data type (optional).
- **name** – A name for the operation (optional).
- **par** – The number of parallel operations (optional).

class nngen.operator.basic.**rshift**(*x*, *y*, *dtype=None*, *name=None*, *par=1*)

Elementwise computes the bitwise right-shift of x and y.

Parameters

- **x** – A tensor.
- **y** – A tensor.
- **dtype** – Output data type (optional).
- **name** – A name for the operation (optional).
- **par** – The number of parallel operations (optional).

class nngen.operator.basic.**rshift_round**(*x*, *y*, *dtype=None*, *name=None*, *par=1*)

Elementwise computes the bitwise right-shift of x and y with rounding.

Parameters

- **x** – A tensor.
- **y** – A tensor.
- **dtype** – Output data type (optional).
- **name** – A name for the operation (optional).
- **par** – The number of parallel operations (optional).

class nngen.operator.basic.**multiply**(*x*, *y*, *dtype=None*, *name=None*, *par=1*)

Returns $x * y$ element-wise.

Parameters

- **x** – A tensor.
- **y** – A tensor.
- **dtype** – Output data type (optional).
- **name** – A name for the operation (optional).
- **par** – The number of parallel operations (optional).

class nngen.operator.basic.**reduce_sum**(*input_tensor*, *axis=None*, *keep_dims=False*, *dtype=None*, *name=None*, *par=1*)

Computes the sum of elements across dimensions of a tensor.

Parameters

- **input_tensor** – A tensor.
- **axis** – The dimensions to reduce (optional).
- **keep_dims** – If true, retains reduced dimensions with length 1 (optional).
- **dtype** – Output data type (optional).
- **name** – A name for the operation (optional).
- **par** – The number of parallel operations (optional).

nngen.operator.basic.**reshape**(*tensor*, *shape*, *dtype=None*, *name=None*)

Reshapes a tensor.

Given tensor, this operation returns a tensor that has the same values as tensor with shape shape.

Parameters

- **tensor** – A tensor.
- **shape** – A tensor. Defines the shape of the output tensor.
- **dtype** – Output data type (optional).
- **name** – A name for the operation (optional).

nngen.operator.basic.**expand_dims**(*input*, *axis*, *name=None*)

Inserts a dimension of 1 into a tensor's shape.

Given a tensor input, this operation inserts a dimension of 1 at the dimension index axis of input's shape. The dimension index axis starts at zero; if you specify a negative number for axis it is counted backward from the end.

Parameters

input – A Tensor.

class nngen.operator.basic.**transpose**(*a*, *perm=None*, *dtype=None*, *name=None*)

Transposes a. Permutes the dimensions according to perm.

Parameters

- **a** – A Tensor.
- **perm** – A permutation of the dimensions of a.
- **dtype** – Output data type (optional).
- **name** – A name for the operation (optional).

class nngen.operator.relu.**relu**(*features*, *dtype=None*, *name=None*, *par=1*)

Applies the rectified linear unit function element-wise:

$$ReLU(x) = \max(0, x)$$

Parameters

- **features** – A Tensor.
- **dtype** – Output data type (optional).
- **name** – A name for the operation (optional).
- **par** – The number of parallel operations (optional).

class nngen.operator.relu.**relu6**(*features*, *dtype=None*, *name=None*, *par=1*)

Applies the element-wise function:

$$ReLU6(x) = \min(\max(0, x), 6)$$

Parameters

- **features** – A Tensor.
- **dtype** – Output data type (optional).
- **name** – A name for the operation (optional).

- **par** – The number of parallel operations (optional).

class nngen.operator.conv2d.**conv2d**(*input*, *filter*, *strides*, *bias=None*, *scale=None*, *rshift_mul=None*, *rshift_sum=None*, *rshift_out=None*, *act_func=None*, *padding='SAME'*, *asymmetric_clip=False*, *dtype=None*, *mul_dtype=None*, *sum_dtype=None*, *name=None*, *par_ich=1*, *par_och=1*, *par_col=1*, *par_row=1*, *concur_och=None*, *stationary='filter'*, *input_ram_size=None*, *filter_ram_size=None*, *bias_ram_size=None*, *scale_ram_size=None*, *vshamt_mul_ram_size=None*, *vshamt_sum_ram_size=None*, *vshamt_out_ram_size=None*, *out_ram_size=None*, *disable_keep_input=False*, *input_shape=None*, *filter_shape=None*, *out_shape=None*)

Computes a 2-D convolution given 4-D input and filter tensors.

Parameters

- **input** – NHWC (batch, height, width, channel)
- **filter** – OHWI (outchannel, height, width, inchannel)
- **stride** – NHWC (N and C are always 1)
- **bias** (*optional*) – Tensor for bias addition to outputs.
- **scale** (*optional*) – Tensor for scaling to outputs.
- **rshift_mul** (*optional*) – Arithmetic shift right can be performed on the result of activation and kernel multiplication as needed. Designation of shift amount can select constant and vector. The constants are register generated and the vector is configured in RAM.
- **rshift_sum** (*optional*) – Arithmetic shift right can be performed on the result of accumulation as needed. Designation of shift amount can select constant and vector. The constants are register generated and the vector is configured in RAM.
- **rshift_out** (*optional*) – Arithmetic shift right can be performed on the output value per channel as needed. Designation of shift amount can select constant and vector. The constants are register generated and the vector is configured in RAM.
- **act_func** (*optional*) – The output value of conv2d can be input to the activation function before writing to memory. The activation function that can be specified is the operator that inherited the element-wise class.
- **padding** (*optional*) – A string (Only 'SAME' padding is supported). The type of padding algorithm to use. The detailed algorithm conforms to Tensorflow.
- **asymmetric_clip** (*optional*) – If this parameter is set to True, the negative range size in clipping operations is increased by one.
- **dtype** (*optional*) – Output data type.
- **mul_dtype** (*optional*) – Data type of register that stores the result of multiplication of activation and kernel parameter.
- **sum_dtype** (*optional*) – Data type of register that stores accumulation result.
- **name** (*optional*) – A name for the operation (optional).
- **par_ich** (*optional*) – Specifies the degree of operation parallelism in the input channel direction.
- **par_och** (*optional*) – Specifies the degree of operation parallelism in the output channel direction.
- **par_col** (*optional*) – Specifies the degree of operation parallelism in the column direction.

- **par_row** (*optional*) – Specifies the degree of operation parallelism in the row direction.
- **concur_och** (*optional*) – Specifies how many channels of kernel parameters to read simultaneously. If this value is large enough, the efficiency of burst transfer in writing the output result is maximized. If not specified, it is calculated automatically from the relationship between the number of words of memory for kernel parameters and the number of words of memory for output.
- **stationary** (*optional*) – Designate data flow to MAC. In 'weight' mode, the weight data supplied to the MAC is fixed, while in 'act' mode, the input data is fixed.
- **input_ram_size** (*optional*) – Specify the word length of the input data RAM. If set to less than the minimum required word length (depends on the input data size), the set value is ignored.
- **filter_ram_size** (*optional*) – Specifies the word length of the kernel parameter RAM. If set to less than the minimum required word length (depends on the kernel parameter size), the set value is ignored.
- **bias_ram_size** (*optional*) – Specify the word length of the bias parameter RAM. If set to less than the minimum required word length (depends on bias parameter size), the set value is ignored.
- **scale_ram_size** (*optional*) – Specify the word length of scaling parameter RAM. If set to less than the minimum required word length (depends on the scaling parameter size), the set value is ignored.
- **vshamt_mul_ram_size** (*optional*) – Specifies the word length of the RAM that stores the arithmetic right shift value for the result of activation and multiplication of kernel parameters. The setting value is ignored if it is set to the minimum required word length or less.
- **vshamt_sum_ram_size** (*optional*) – Specifies the word length of the RAM that stores the arithmetic right shift value for the accumulation result. The setting value is ignored if it is set to the minimum required word length or less.
- **vshamt_out_ram_size** (*optional*) – Specifies the word length of the RAM that stores the arithmetic right shift value for the output result. The setting value is ignored if it is set to the minimum required word length or less.
- **out_ram_size** (*optional*) – Specifies the word length of the output data RAM. If set to less than the minimum required word length (depends on the output data size), the set value is ignored.
- **disable_keep_input** (*optional*) – If this parameter is set to True, data will be reread from the main memory even if the required amount of input data is full on the on-chip RAM.

Notes

Note that the original order of tensorflow's conv2d is `HWIO` (height, width, inchannel, outchannel)

collect_local_control_param_values(*index_offset=0*)

conv2d has no local_control_params, but self.act_func can have local_control_params. So collect_local_control_param_values() returns those of act_func.

control_sequence(*fsm*)

must be implemented in each _Operator class

copy_local_control_params(*obj*, *index_offset=0*)

conv2d has no local_control_params, but self.act_func can have local_control_params. So all local_control_params objects are assigned to act_func object.

get_control_param_values()

This method must be implemented in each _Operator class. For chained _StreamingOperator, this method of the head operator only is called. If control params of intermediate operators in the chain are required, use get_local_control_param_values()

get_required_rams()

@return 3 tuples of (width, length)

get_required_substreams()

@return a tuple of (method_name, args)

get_stream_func()

must be implemented in each _Operator class

CHAPTER

TWO

PYNQ

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

INDEX