2023 Digital IC Design Homework 4

NAME	林珮玉
Student ID	E24084096

Simulation Result

Functional simulation	100	Gate-level simulation	100

```
# ----------------------------------------------------
#
# START!!! Simulation Start .....
#
# ----------------------------------------------------
#
# Layer 0 output is correct !
# Layer 1 output is correct!
#
# ----------------------------------------------------
#
# ---------------------- S U M M A R Y -----------------
#
# Congratulations! Layer 0 data have been generated successfully! The result is PASS!!
#
# Congratulations! Layer 1 data have been generated successfully! The result is PASS!!
#
# terminate at      19464 cycle
# ----------------------------------------------------
#
# ** Note: $finish    : C:/DIC/HW4/file/testfixture.v(178)
#    Time: 973200 ns  Iteration: 0  Instance: /testfixture
```

```
# ----------------------------------------------------
#
# START!!! Simulation Start .....
#
# ----------------------------------------------------
#
# Layer 0 output is correct !
# Layer 1 output is correct!
#
# ----------------------------------------------------
#
# ---------------------- S U M M A R Y -----------------
#
# Congratulations! Layer 0 data have been generated successfully! The result is PASS!!
#
# Congratulations! Layer 1 data have been generated successfully! The result is PASS!!
#
# terminate at      19464 cycle
# ----------------------------------------------------
#
# ** Note: $finish    : C:/DIC/HW4/file/testfixture.v(178)
#    Time: 973200 ns  Iteration: 0  Instance: /testfixture
```

Synthesis Result

Total logic elements	
Total memory bits	
Embedded multiplier 9-bit elements	
Total cycle used	

(your flow summary)

Description of your design

The Verilog code provided is an implementation of a convolutional neural network (CNN) accelerator module called ATCONV. Here is a description of the various components and functionalities of the module:

Inputs:

clk: Clock input signal.
reset: Reset input signal.
ready: Ready input signal.

Outputs:

busy: Busy output signal.
iaddr: Address output signal for input data.
idata: Signed input data.
cwr: Control signal for convolution data write.
caddr_wr: Address output signal for convolution data write.
cdata_wr: Data output signal for convolution data write.
crd: Control signal for convolution data read.
caddr_rd: Address output signal for convolution data read.
cdata_rd: Data input signal for convolution data read.
csel: Control signal for selecting between convolution and pooling.
Internal Signals and Variables:

state, nextState: Registers for representing the current and next state of the state machine.
layer0, layer0_, layer1: Registers representing memory arrays for storing intermediate results of the CNN layers.
index: Register for keeping track of the current index in the memory arrays.
idx, idy: Registers for indexing and iterating over the input data and intermediate layers.
en, en1: Control signals for enabling certain operations.
i, j: Loop iterators.
State Parameters:

READY: Initial state, waiting for the ready signal to be asserted.
IMAGE_MEM: State for storing input data into the layer0 memory array.
PADDING: State for performing padding on the input data stored in layer0.
CONVOLUTION: State for performing convolution on the padded input data.
RELU: State for applying the Rectified Linear Unit (ReLU) activation function on the convolved data.
LAYER_1_WRITE: State for storing the ReLU-activated data into the layer0 memory.
MAX_POOLING: State for performing max pooling on the stored data in layer0 and storing the result in layer1.
ROUND_UP: State for rounding up the values in layer1 to 13 bits.
LAYER_2_WRITE: State for storing the rounded-up data into the layer1 memory.
WAIT: Final state, waiting for the next computation to start.

Functionality:

The module uses a state machine to control the execution flow of the CNN accelerator.
It starts in the READY state and waits for the ready signal to be asserted.
Once the ready signal is asserted, the module transitions to the IMAGE_MEM state and starts storing the input data into the layer0 memory.
After storing all the input data, the module enters the PADDING state, where padding is applied to the input data in layer0.
Next, the module enters the CONVOLUTION state and performs convolution on the padded input data using predefined weights and biases.
In the RELU state, the ReLU activation function is applied to the convolved data.
The module then transitions to the LAYER_1_WRITE state, where the ReLU-activated data is stored back into the layer0 memory.
Afterward, in the MAX_POOLING state, max pooling is performed on the stored data in layer0, and the result is stored in layer1.
The module enters the ROUND_UP state, where the values in layer1 are rounded up to 13 bits

*Scoring = (Total logic elements + Total memory bits + 9*Embedded multipliers 9-bit elements) X Total cycle used*

*** Total logic elements must not exceed 1000.**