2023 Digital IC Design

Homework 4: Atrous Convolution

1. Introduction:

Atrous convolution is a technique that expands the kernel by inserting holes between its consecutive elements. In simpler terms, it is a special type of convolution that involves pixel skipping to expand the receptive field without increasing parameters. For example, a 3x3 atrous convolution kernel with the hyperparameter **dilation**=2 can have the same receptive field as a 5x5 regular convolution kernel. By altering the hyperparameter **dilation**, atrous convolution can capture features at multiple scales. Moreover, atrous convolution can reduce the spatial resolution loss compared with regular convolution. Atrous convolutions have been used successfully in various applications, such as semantic segmentation, where a larger context is needed to classify each pixel, and audio processing, where the network needs to learn patterns with longer time dependencies.

In this homework, you are requested to design a two-layered atrous convolutional circuit. The effect of the atrous convolution layer is shown in Figure 1. The input image size is fixed as 64x64. Layer 0 consists of padding, atrous convolution, and ReLU operations. While Layer 1 scales the feature map from Layer 0 to the size of the output image (32x32) with the max-pooling operation, and rounds up the results to the nearest integers. More detailed circuit functionalities will be described in the subsequent sections.

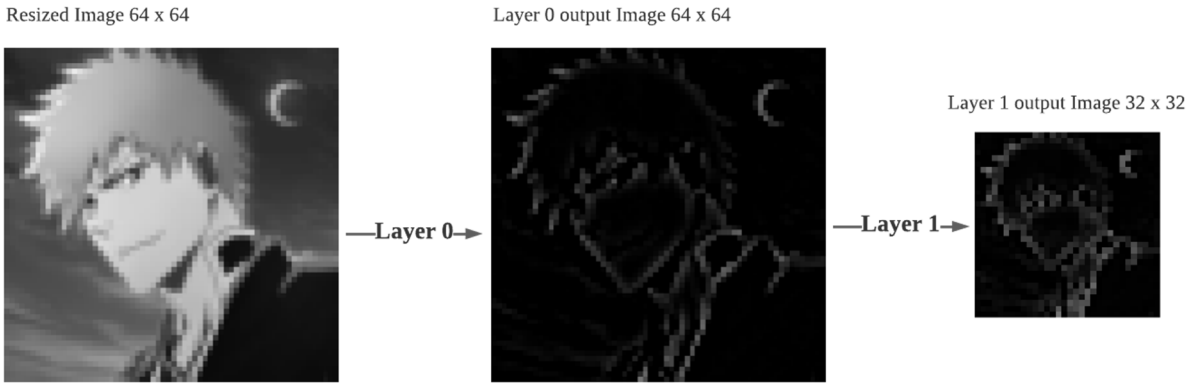


Figure 1. Atrous convolution example.

2. Design Specifications:

2.1 Block Overview:

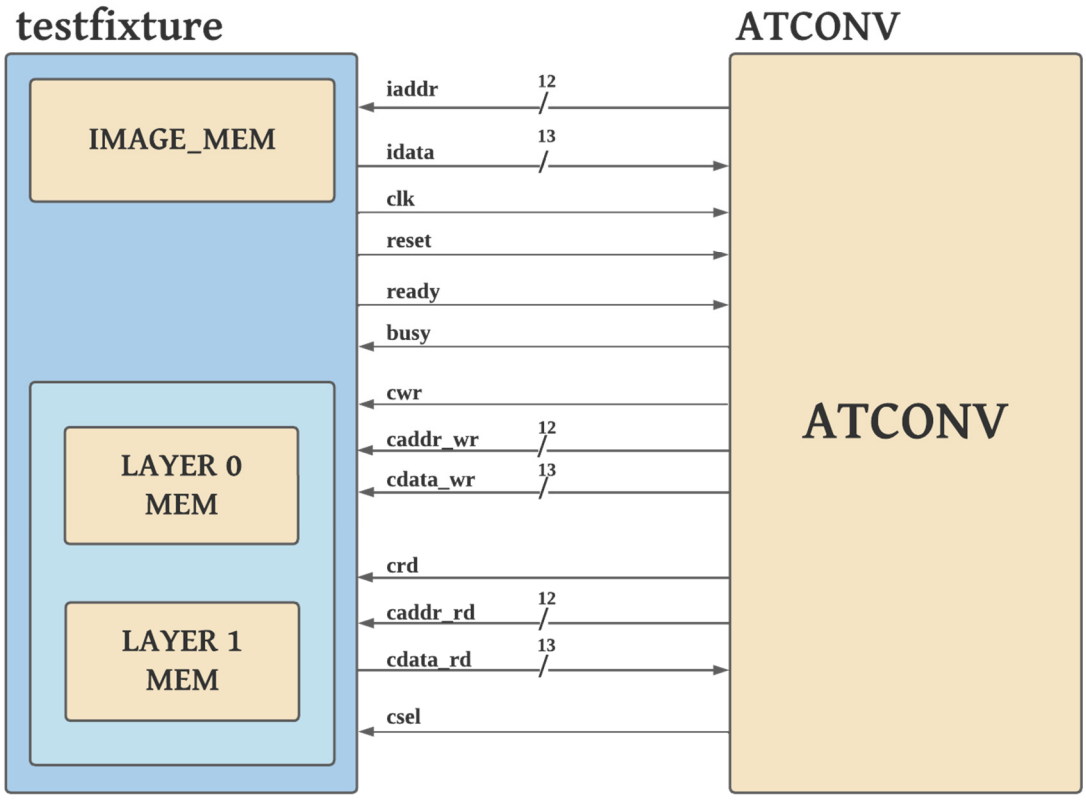


Figure 2. System Block Diagram

2.2 I/O Interface:

Signal Name	I/O	width	Description
clk	I	1	System clock signal. This system is synchronized at the positive edge of the clock.
reset	I	1	Active-high asynchronous reset signal.
ready	I	1	Image memory ready signal. After the 64x64 grayscale image has already been loaded to the IMAGE_MEM, the *ready* signal is pulled up to high. ATCONV circuit can start requiring valid *idata* by assigning corresponding *iaddr* after IMAGE_MEM is ready. The *ready* signal will be pulled down when the testbench detects the *busy* signal is high.

busy	O	1	After *ready* signal is set as high, ATCONV circuit should pull the *busy* signal to high. After ATCONV circuit finishing all the tasks, pull the *busy* signal to low, and the testfixture will check the result.
iaddr	O	12	Image memory address signal. Specifying the address of the requested grayscale image pixel.
idata	I	13	Input pixel data of the grayscale image. The pixel data consists of 9-bit integer part (MSB) and 4-bit fractional part (LSB). The testfixture will send the data whose memory address is *iaddr* with *idata* port to ATCONV circuit.
csel	O	1	Memory selection signal. When csel signal is low, Layer 0 memory is selected. When csel signal is high, Layer 1 memory is selected. ATCONV circuit can only read/write the selected memory.
crd	O	1	Read enable signal. If *crd* signal is set as high, the data at the address *caddr_rd* of the selected memory will be read and transmitted to ATCONV circuit with *cdata_rd* port.
caddr_rd	O	12	Read address signal. Specifying the request data address of the selected memory.
cdata_rd	I	13	Read data signal. The data read from the selected memory would be transmitted to ATCONV with *cdata_rd* port.
cwr	O	1	Write enable signal. If *cwr* signal is set as high, the value of *cdata_wr* will be written to the address *caddr_wr* of the selected memory.
caddr_wr	O	12	Write address signal. Specifying the address of the selected memory to be written.
cdata_wr	O	13	Write data signal. The data to be written to the address *caddr_wr* of the selected memory.

Table I. I/O interface of the design

2.3 Function Description:

2.3.1 Atrous convolutional circuit behavior overview

Input image is a 64 x 64 grayscale image stored in the IMAGE_MEM.

In Layer 0, there are three main tasks to accomplish.

1. Replicate padding the 64x64 grayscale input image to 68x68. This operation enables the output feature map of atrous convolution to maintain the same image size.
2. Apply atrous convolution to the padded grayscale image
3. Implement ReLU function on the result of astrous convolution

The result of Layer 0 should be saved in Layer 0 MEM.

In Layer 1, there are two main tasks to accomplish.

1. Max-pooling the output of Layer 0 (stride=2, kernel_size=2x2)
2. Round up the result of Max-pooling to the nearest integer

The result of Layer 1 should be saved in Layer 1 MEM.

The testfixture would check the values in Layer 0 MEM and Layer 1 MEM after *busy* signal is set to low.

Notice：The input data and golden results of Layer 0 and Layer 1 all consist of 9-bit integer part (MSB) and 4-bit fractional part (LSB). However, you may need to use register larger than 13 bits to store the intermediate calculation results.

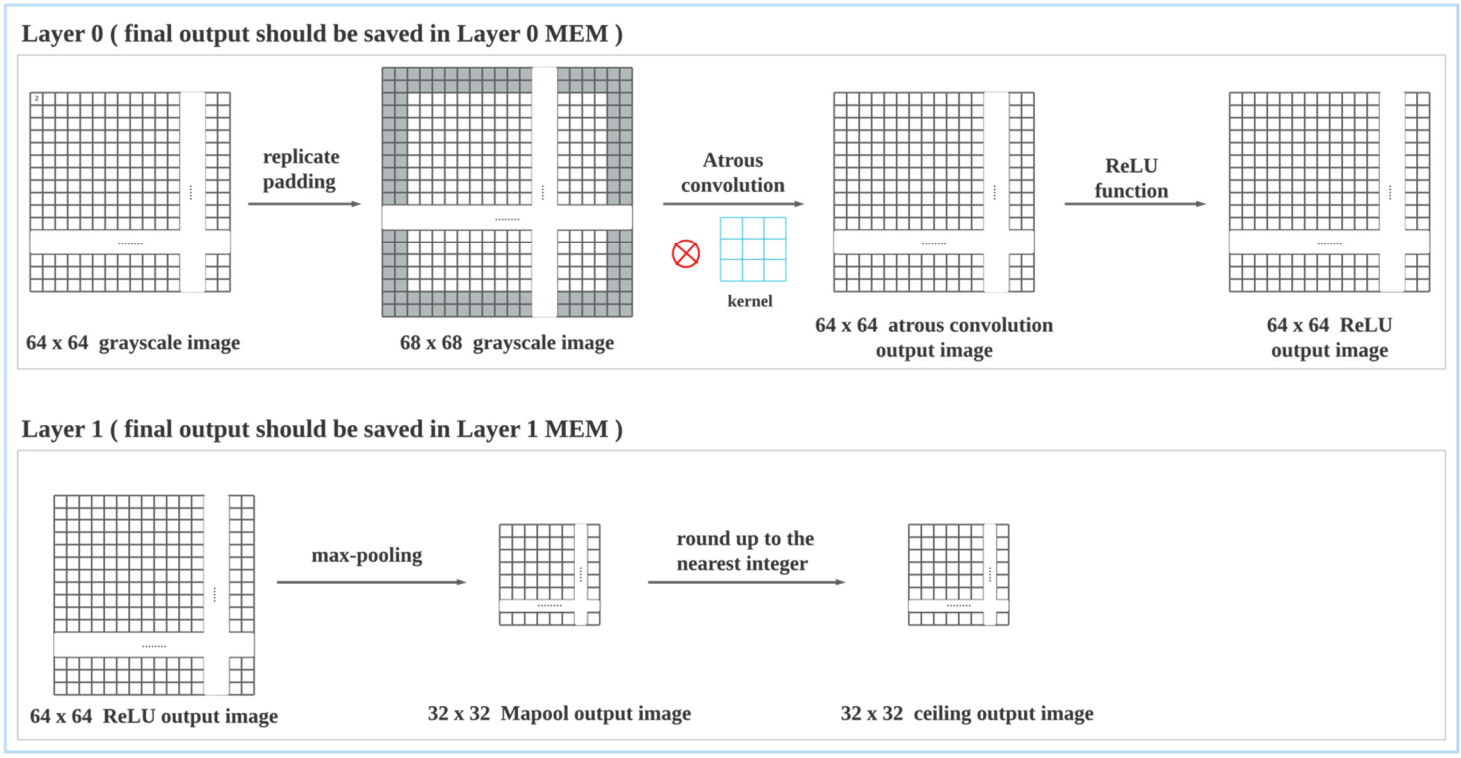


Figure 3. Atrous convolutional circuit behavior overiew

2.3.2 Memory Mapping Relations

1. Relation between input grayscale image and IMAGE_MEM

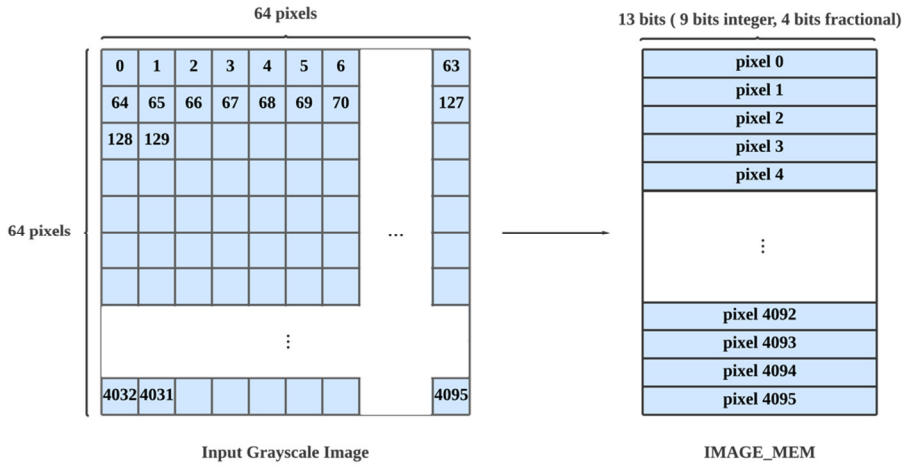


Figure 4. Input Grayscale Image memory mapping

2. Relation between Layer 0 results and Layer 0 MEM

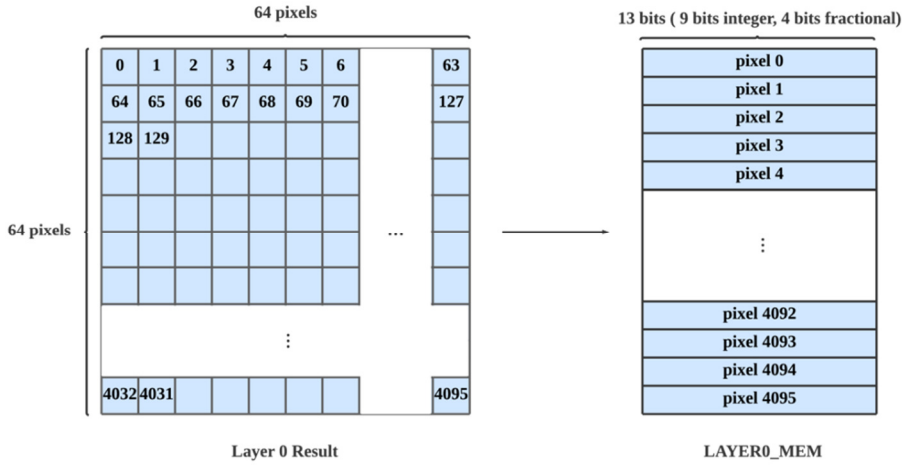


Figure 5. Layer 0 result memory mapping

3. Relation between Layer 1 results and Layer 1 MEM

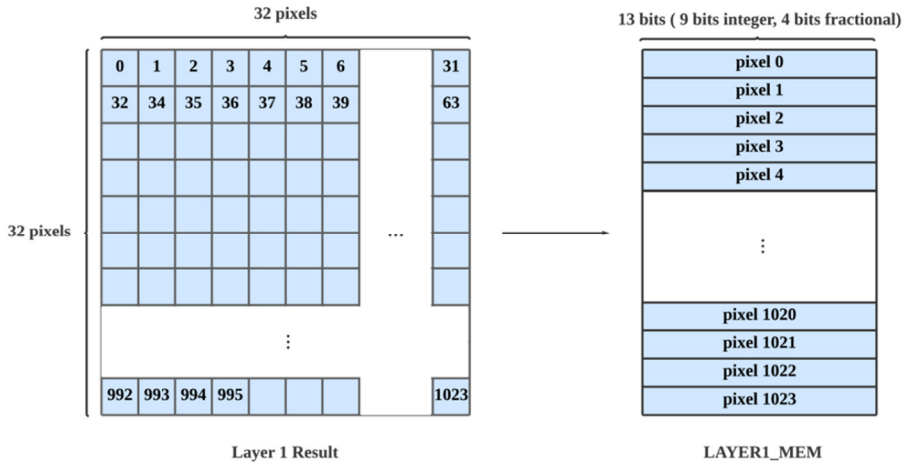


Figure 6. Layer 1 result memory mapping

2.3.3 Replicate padding operation:

To maintain the same image size (64x64) after atrous convolution, replicate padding the input grayscale image to 68x68 is required. Below is an example of how replicate padding works on the grayscale image. When the padding parameter is 1, the values at the original boundary would be replicated to form the new boundary. When the padding parameter is 2, the replication operation will be conducted two times to create two new pixels at each side on the boundary. The 3x3 image in the example becomes 7x7 after the padding operation with padding parameter is 2.

In this assignment, the padding parameter is set as 2. The 64x64 image will become 68x68 after the replicate padding operation.

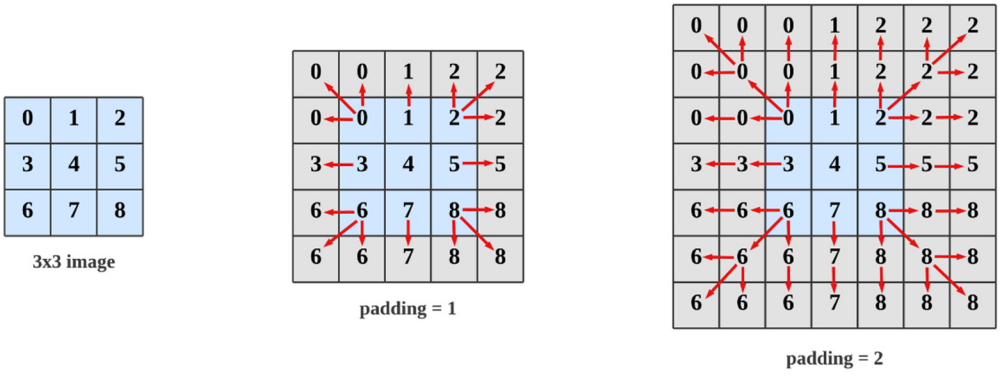


Figure 7. replicate padding example

2.3.4 Atrous Convolution:

The hyperparameter **dilation** controls the spacing between the convolved points. The atrous convolution with **dilation** is 1 is the same as the regular convolution. The hyperparameter **stride** indicates the step of shifting window, the shifting window will move by *n* pixels when **stride** is *n*.

For the atrous convolution operation in this homework, the hyperparameter **dilation** is set as 2 and **stride** is set as 1. The following example will use a 3x3 kernel as an illustration, with the hyperparameter **dilation** is 2 and **stride** is 1. The atrous convolution would be implemented on the padded image of size 7x7.

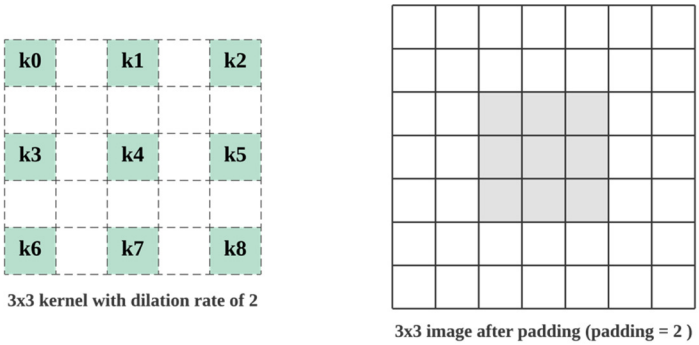
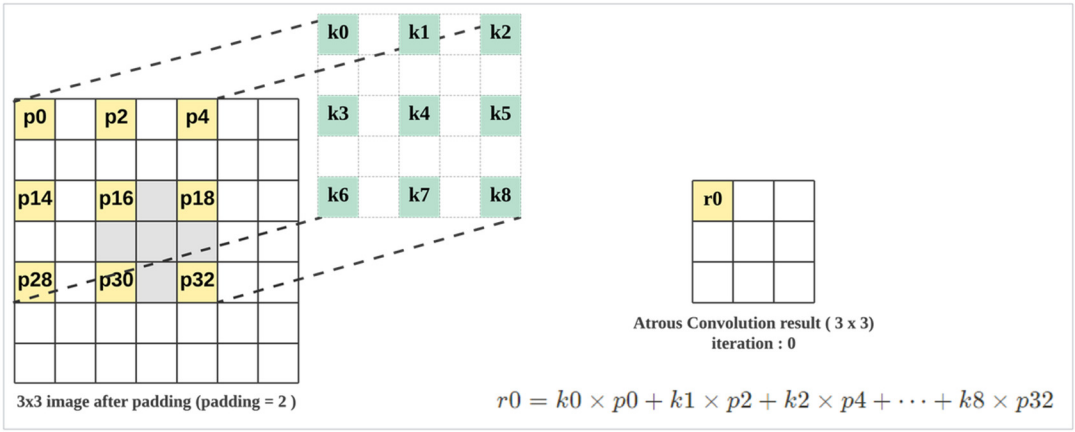
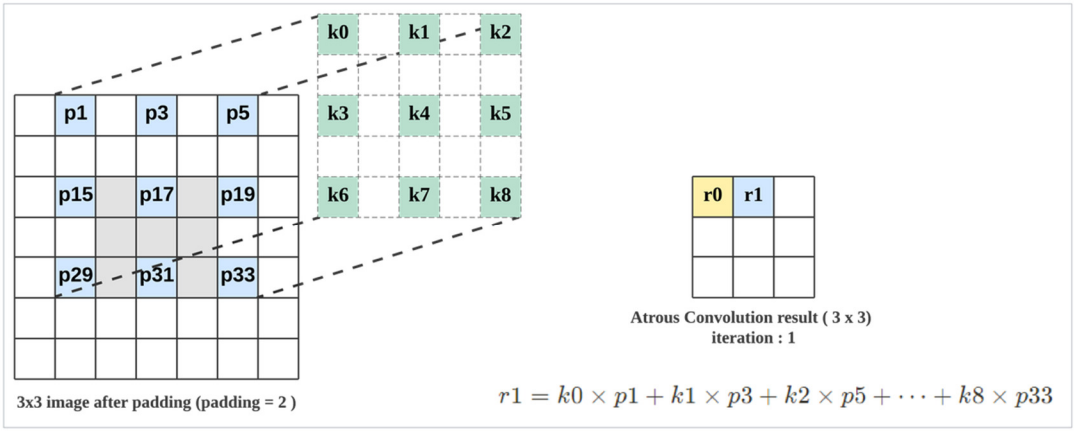
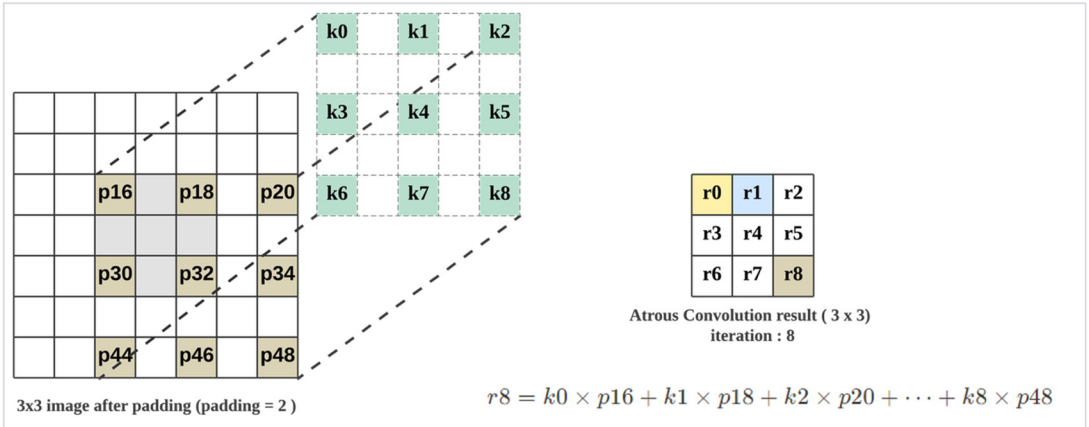


Figure 8. Kernel and image used in example







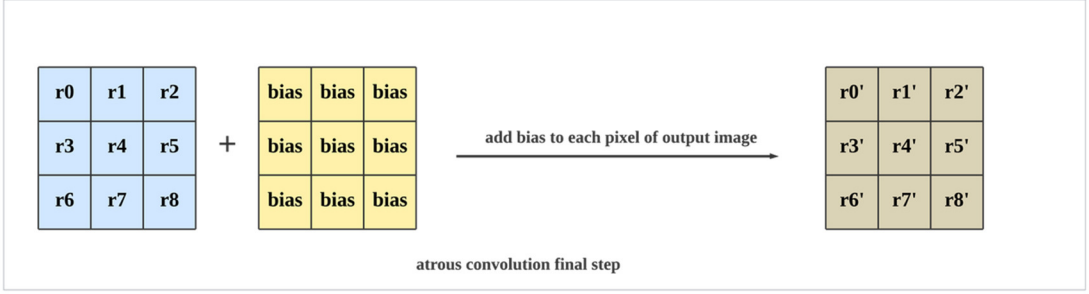


Figure 9. Atrous convolution example on 7x7 padded image (iteration 0, 1, 8)

Figure 9 shows the atrous convolution process, which will scan the whole image from left to right, from top to bottom with a shifting window. The pixel values in the window will convolve with the values of the kernel. From the example in Figure 9, it can be seen the output of atrous convolution has the same size (3x3) as the original input image, which result from the padding operation. Figure 10 shows the kernel values and bias value used in this homework. The kernel values and bias value consist of 9-bit integer part (MSB) and 4-bit fractional part (LSB), and the MSB represents sign bit.

Figure 10. Kernel and bias used in this assignment

2.3.5 ReLU Function:

In the last step of Layer 0, ReLU function is applied to the output of atrous convolution. The definition of ReLU function is shown below, which assigns 0 to the non-positive value.

$$y = \begin{cases} x\ (x > 0) \\ 0\ (x \leq 0) \end{cases}$$

2.3.6 max-pooling Example:

Max-pooling is adopted in the first step of Layer 1. How max-pooling works is shown in Figure 11 below. In this homework, the kernel size of max-pooling is 2x2 and the hyperparameter **stride** of max-pooling is 2. The example below adopts the same hyperparameters used in this homework.

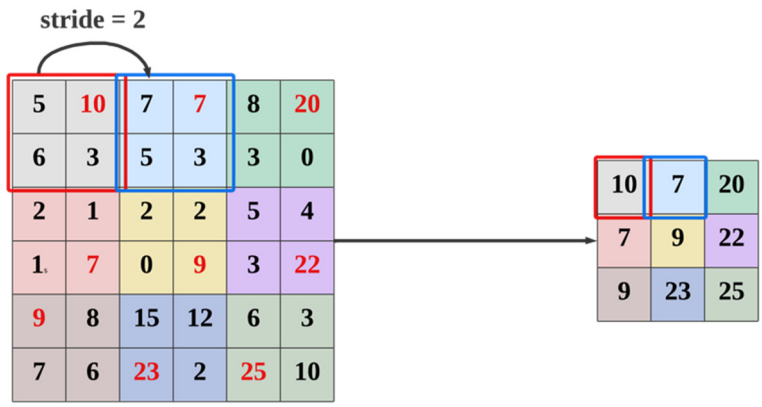


Figure 11. Max-pooling example

2.4. Timing Specifications:

2.4.1 ATCONV system control

When the input grayscale image is ready (***ready*** signal is high), ATCONV circuit should pull the ***busy*** signal to high before further action. The ***ready*** signal will be pulled down after the testbench detects ***busy*** signal is high. When ATCONV circuit finish all the tasks, pull the ***busy*** signal as low to inform the testfixture to validate the results.

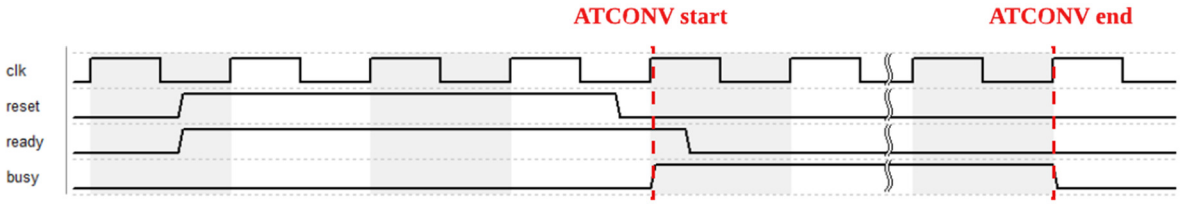


Figure 12. system busy control

2.4.2 IMAGE_MEM data reading control

After the input grayscale image is ready, ATCONV circuit can retrieve data with ***iaddr*** and ***idata*** ports. Assign the address of requiring data to ***iaddr*** at the positive edge of ***clk***, and the retrieved data will be transmitted to ATCONV circuit at the negative edge of ***clk***. Therefore, ATCONV circuit can get the retrieved data at the next positive edge of ***clk***.

Figure 13. input image pixel retrieval control

2.4.3 Layer 0/1 MEM data reading control

When the ***crd*** signal is set to 1, ATCONV circuit can read data from the memory selected with ***csel*** signal. After assign the data address with ***caddr_rd*** port at the positive edge of ***clk***, the testfixure would transmit the corresponding data to the ***cdata_rd*** port at the negative edge of clk. Therefore, ATCONV circuit can get the data at the next positive edge of ***clk***.

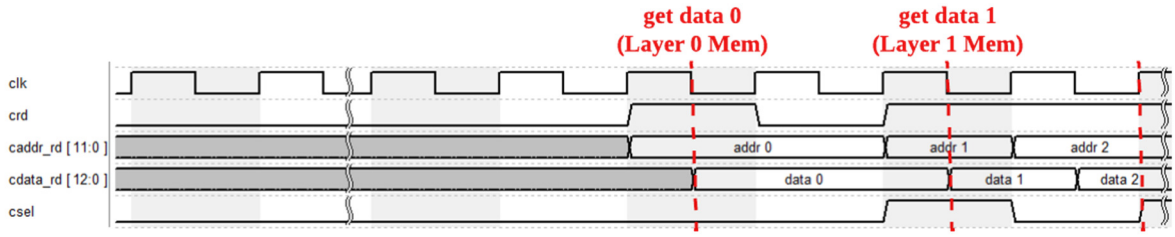


Figure 14. memory data reading control

2.4.4 Layer 0/1 MEM data writing control

When the ***cwr*** signal is set to 1, ATCONV circuit can write data into the memory selected with ***csel*** signal. Assign the data address and the data to ***caddr_wr*** and ***cdata_wr*** at the positive edge of ***clk***, the data would be written to the corresponding address of the selected memory at the negative edge of ***clk***.

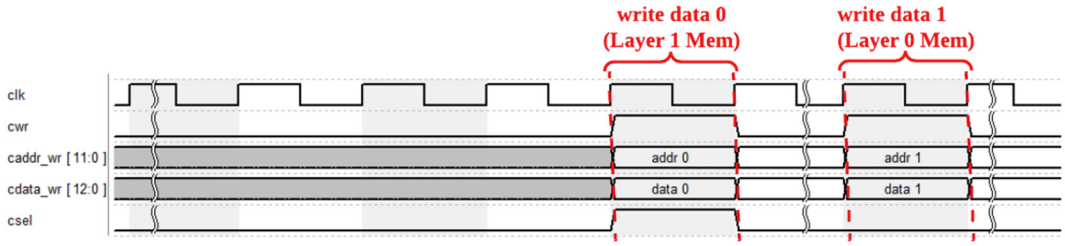


Figure 15. memory data writing control

3. Software Verification:

In this homework, you need to write a software program to verify your circuit. You can write your program using C, C++ and python (python is recommended). The requirements and functionality of the software program are as following:

1. Be able to read any .jpg or .png image. The file path for reading should be ./image.jpg or ./image.png.
2. Convert RGB image to grayscale.
3. Resize the grayscale image into 64x64 and output the resized image as img.dat.
4. Implement the algorithm of layer 0 and output the result of layer 0 as layer0_golden.dat.
5. Implement the algorithm in layer 1 and output the result of layer 1 as layer1_golden.dat.

6. The format of the three output files should be the same as the img.dat, layer0_golden.dat and layer1_golden.dat which TA provided. The output files should be able to be read by the testbench and simulate correctly.
   (hint: you should transform IEEE754 floating point into fixed point number and save it using binary representation.)
7. Please name your program in the format: main.c, main.cpp or main.py.
8. If your program needs to be executed with special libraries or in special environment, please provide a Readme.txt to explain how to execute it.

(Notice: pytorch is allowed to be used, the behavior should be the same as your Verilog code.)

4. File Description:

File Name	Description
ATCONV.v	The top module of your design.
testfixture.v	The testbench file. The content in this file is not allowed to be modified.
img.dat	Input grayscale image data.
layer0_golden.dat	Golden data of layer 0 output results.
layer1_golden.dat	Golden data of layer 1 output results.

5. Scoring:

5.1 Functional simulation [60%]

All of the result should be generated correctly, and you will get the following message in ModelSim simulation. The functional simulation results of Layer 0 and Layer 1 take 30% of the score each.

```
# ------------------------------------------------------
#
# 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      87046 cycle
# ------------------------------------------------------
#
# ** Note: $finish    : F:/master1/DicHomework/hw4/gateSimNew/testfixture.v(179)
```

Figure 16. functional simulation result example

5.2 Gate-Level Simulation [20%]

5.2.1 Synthesis:

Your code should be synthesizable. After it is synthesized in Quartus, a file named ATCONV.vo will be obtained.

DEVICE：Cyclone IV E - EP4CE55F23A7

5.2.2 Simulation:

All of the results should be generated correctly using ATCONV.vo, and you will get the following message in ModelSim simulation. In this homework, the clock width is fixed at 50 ns. The synthesized circuit has to be able to pass simulation with the specified clock width. The gate-level simulation results of Layer 0 and Layer 1 take 10% of the score each.

```
# ----------------------------------------------------
#
# 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      87046 cycle
# ----------------------------------------------------
#
# ** Note: $finish    : F:/master1/DicHomework/hw4/modelsimRoot/testfixture.v(179)
```

Figure 17. gate level simulation result example

5.3 Performance [10%]

The performance is scored by the total logic elements, total memory bits, and embedded multiplier 9-bit elements your design used in gate-level simulation and the total cycles your design takes to finish the simulation.

The performance score will be decided by your ranking in all received homework. Only designs that passed gate-level simulation and meet resource limitations will be considered in the ranking. Otherwise, you can't get performance score.

The scoring standard: (The smaller, the better)

Scoring = (Total logic elements + total memory bits + 9*embedded multiplier 9-bit elements) × (Total cycle)

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

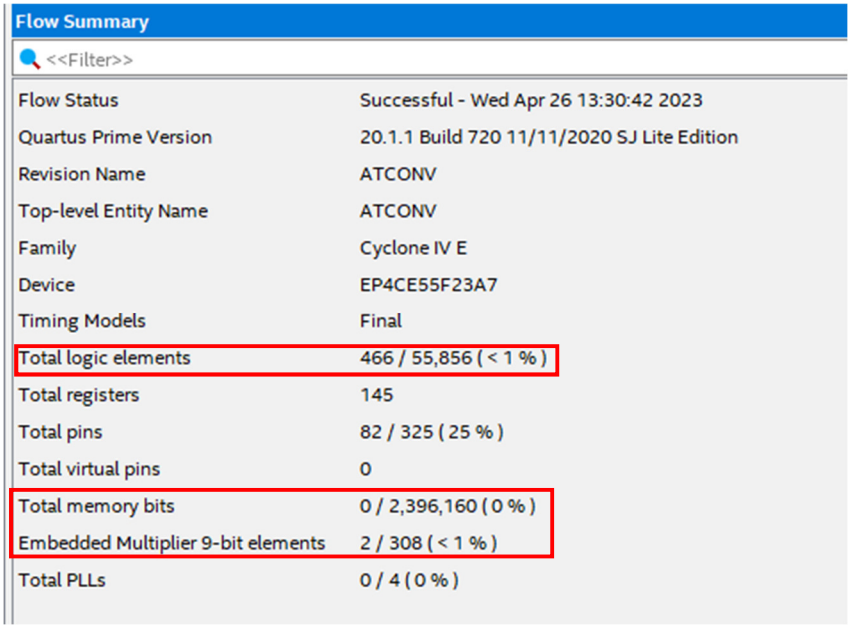


Figure 18. Quartus synthesis result example

5.4 Software Verification [10%]

All requirements have to be completed. TA will generate testing data with your software program to verify your design. Please ensure your design can pass with any testing data generated by your software program. Note that TA will check your program and circuit design with hidden test pattern. You will not get any score of software verification if you fail to pass the hidden test pattern.

6. Submission:

6.1 Submitted files

You should classify your files into three directories and compress them to .zip format. The naming rule is HW4_studentID_name.zip. If your file is not named according to the naming rule, you will lose five points.

	RTL category
*.v	All of your Verilog RTL code
	Gate-Level category
*.vo	Gate-Level netlist generated by Quartus
*.sdo	Gate-Level netlist generated by Quartus
	Software category
*.c/.cpp/.py	Software verification program
Readme.txt	Explain how to execute your software program.

	Documentary category
*.pdf	The report file of your design (in pdf).

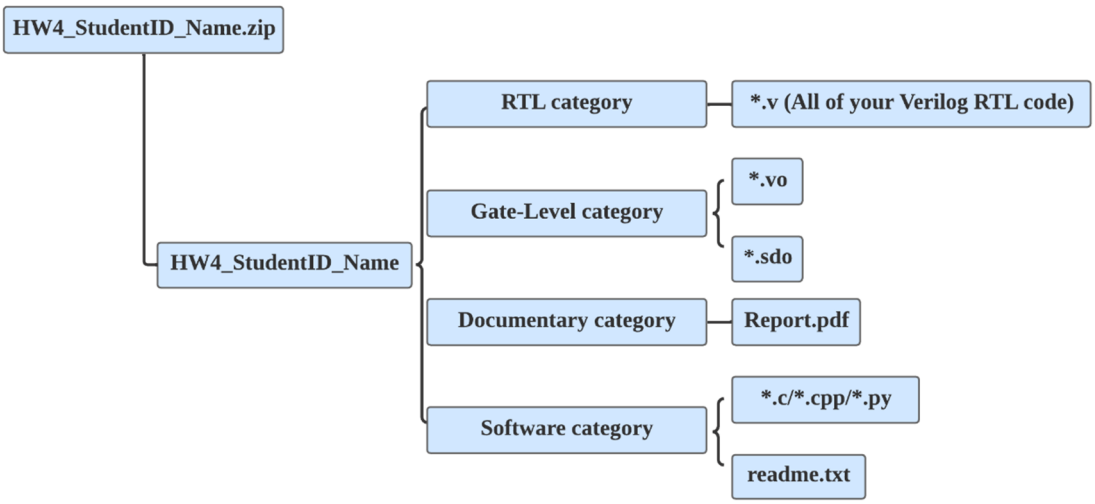


Figure 19. assignment directory

6.2 Report file

Please follow the spec of report. You are asked to describe how the circuit is designed as detailed as possible, and the flow summary result is necessary in the report. Please fill the field of total logic elements, total memory bits, embedded multiplier 9-bit elements according to the flow summary (as shown in Figure 18) of your synthesized design. And fill the field of total cycle used according to the gate-level simulation results that Modelsim shows. If the filled-in values don't match your synthesized design or gate-level simulation results, you will lose 10 points.

6.3 Note

Please submit your .zip file to folder HW4 in moodle.
Deadline: 2023/05/22 23:55

Please don't design specifically for the test pattern. Otherwise, you will get 0 point.
Any plagiarism behavior will also get 0 point.

If you have any problem, please contact TA by email
P76114537@gs.ncku.edu.tw