2025 Digital IC Design

Homework 4: Atrous Convolution with Bus Interface

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.

After finishing the atrous convolution circuit, an AXI-liked bus interface is asked to be design to connect the atrous convolution circuit with other three slave interfaces. The system contains a master interface, a 16-bit AXI-liked bus, and three slave interfaces which is one image ROM and two SRAM.

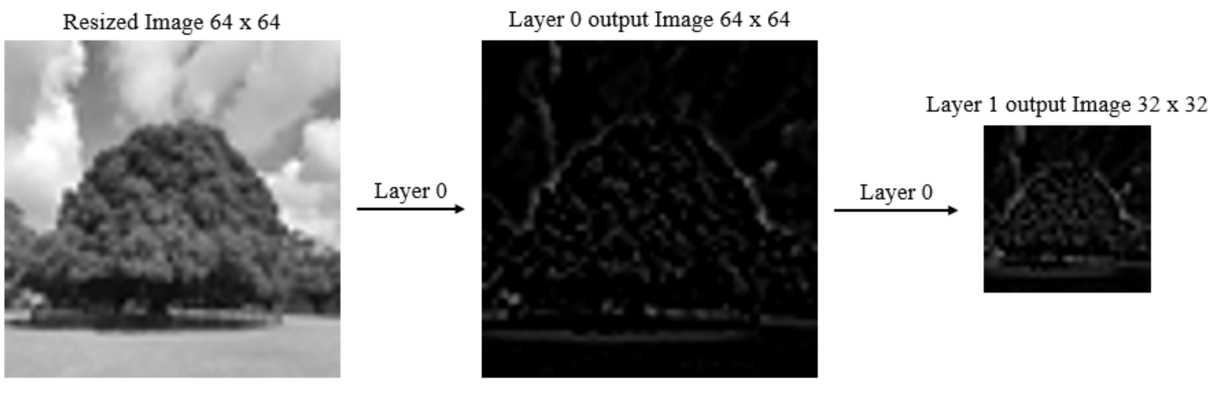


Figure 1. Atrous convolution example.

2. ATCONV Design Specifications:

2.1 Block Overview:

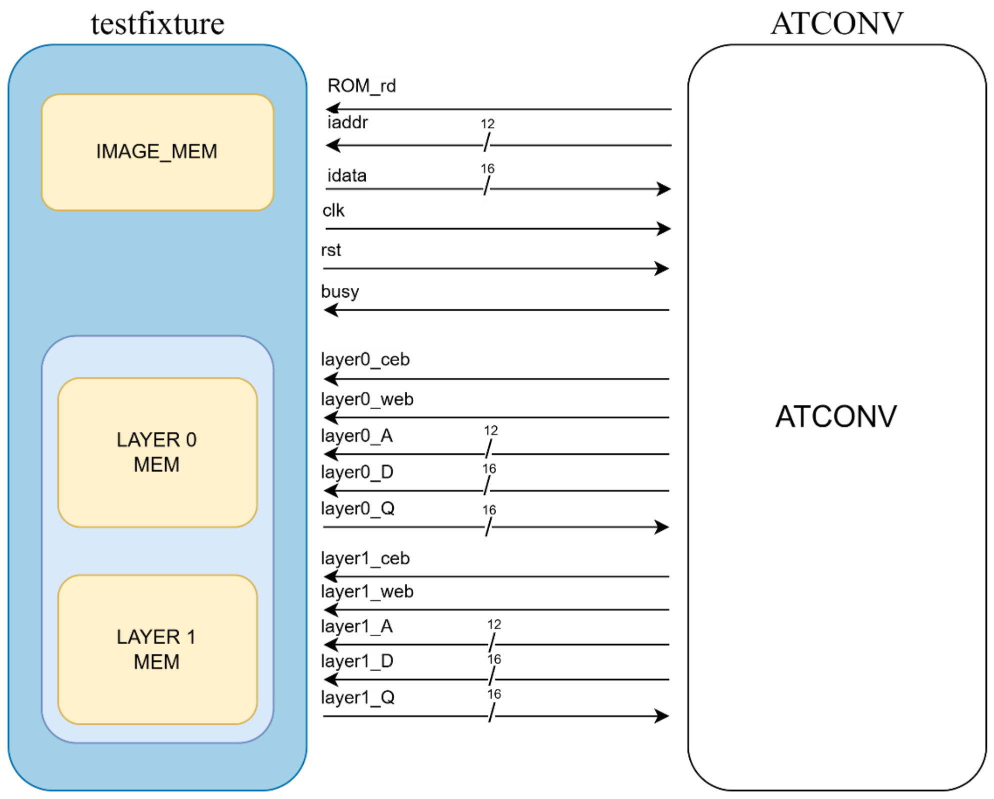


Figure 2. Block Diagram

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.
rst	I	1	Active-high asynchronous reset signal.

ROM_rd	O	1	Image memory read enable signal. When high, indicates the ATCONV requests data from the Host.
iaddr	O	12	Image memory address signal. Specifying the address of the requested grayscale image pixel.
idata	I	16	Input pixel data of the grayscale image. The pixel data consists of 12-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.
layer0_ceb	O	1	LAYER_0_MEM chip enable signal. When high, indicates the LAYER_0_MEM is available for read and write operations.
layer0_web	O	1	LAYER_0_MEM read/write select signal. When High, it indicates that ATCONV is reading data from LAYER_0_MEM; otherwise, it indicates writing data to LAYER_0_MEM.
layer0_A	O	12	LAYER_0_MEM address bus. The ATCONV uses this bus to specify the address in the Host's LAYER_0_MEM where the grayscale image data should be read or written.
layer0_D	O	16	LAYER_0_MEM input data bus. The ATCONV uses this bus to write image data into the Host's LAYER_0_MEM.
layer0_Q	I	16	LAYER_0_MEM output data bus. The Host's LAYER_0_MEM uses this bus to output image data to the ATCONV.
layer1_ceb	O	1	LAYER_1_MEM chip enable signal. When high, indicates the LAYER_1_MEM is available for read and write operations.
layer1_web	O	1	LAYER_1_MEM read/write select signal. When High, it indicates that ATCONV is reading data from LAYER_1_MEM; otherwise, it indicates writing data to LAYER_1_MEM.
layer1_A	O	12	LAYER_1_MEM address bus. The ATCONV uses this bus to specify the address in the Host's LAYER_1_MEM where the grayscale image data should be read or written.

layer1_D	O	16	LAYER_1_MEM input data bus. The ATCONV uses this bus to write image data into the Host's LAYER_1_MEM.
layer1_Q	I	16	LAYER_1_MEM output data bus. The Host's LAYER_1_MEM uses this bus to output image data to the ATCONV.
done	O	1	When the ATCONV completes writing to LAYER_0_MEM and LAYER_1_MEM, it sets done to high to indicate completion.

Table I. I/O interface of ATCONV

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.

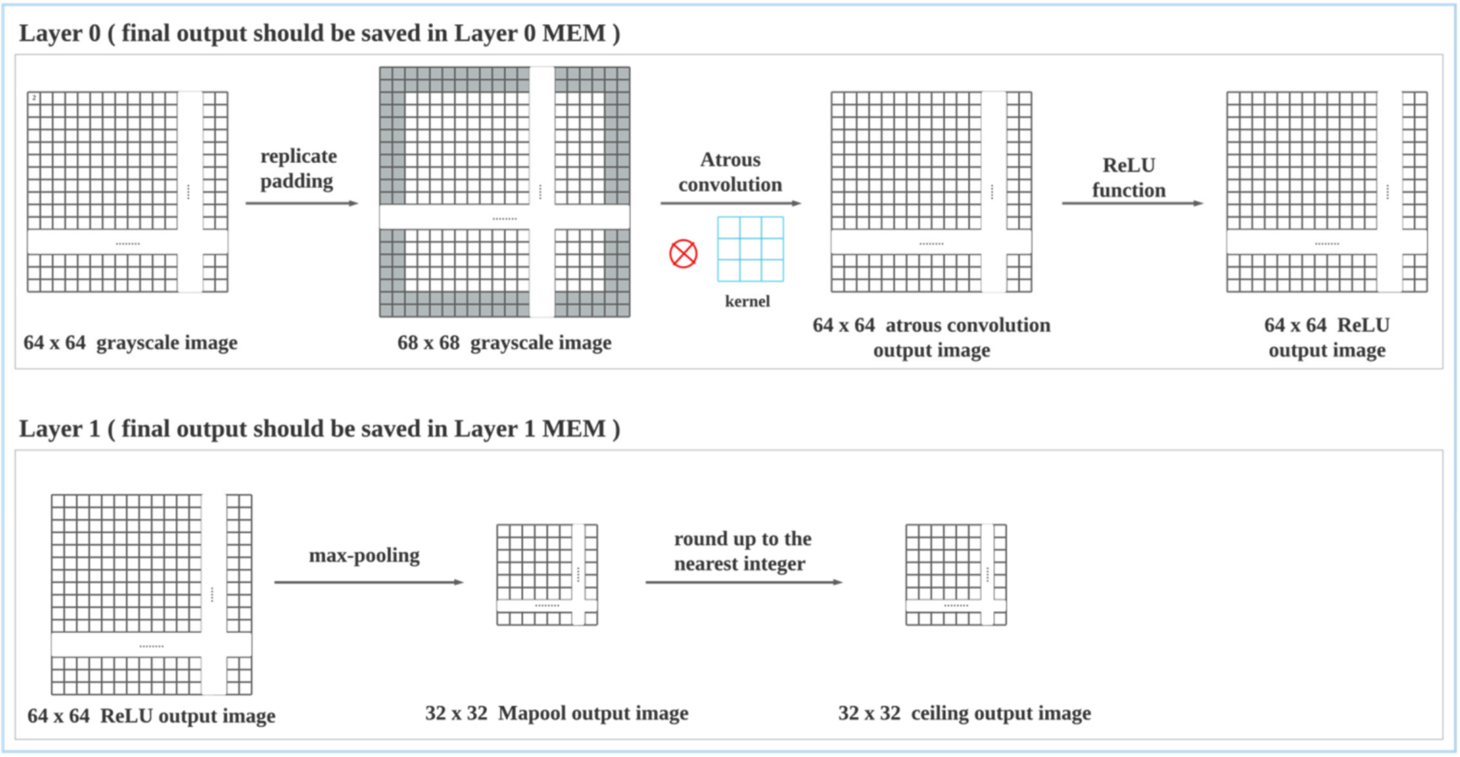
In Layer 1, there are two main tasks to accomplish.

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 ***done*** signal is set to high.

Notice：The input data and golden results of Layer 0 and Layer 1 all consist of 12-bit integer part (MSB) and 4-bit fractional part (LSB). However, you may need to use register larger than 16 bits to store the intermediate calculation results.



Layer 0 (final output should be saved in Layer 0 MEM)
replicate padding
Atrous convolution
kernel
ReLU function
64 x 64 grayscale image
68 x 68 grayscale image
64 x 64 atrous convolution output image
64 x 64 ReLU output image
Layer 1 (final output should be saved in Layer 1 MEM)
max-pooling
round up to the nearest integer
64 x 64 ReLU output image
32 x 32 Mapool output image
32 x 32 ceiling output image

Figure 3. Atrous convolutional circuit behavior overiew

2.3.2 Memory Mapping Relations

1. **Relation between input grayscale image and IMAGE_MEM**

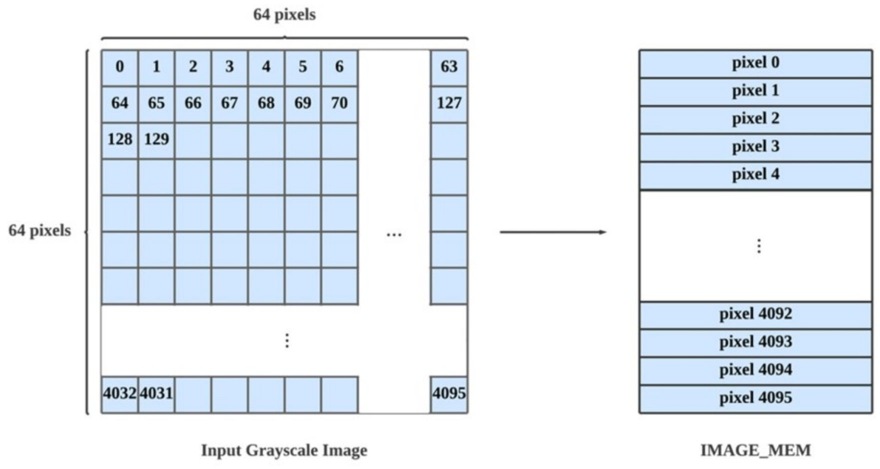


Figure 4. Input Grayscale Image memory mapping

2. **Relation between Layer 0 results and Layer 0 MEM**

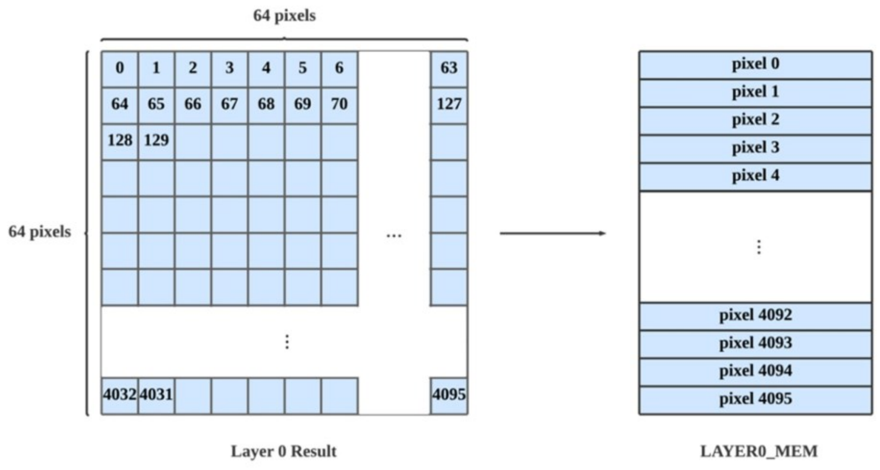


Figure 5. Layer 0 result memory mapping

3. **Relation between Layer 1 results and Layer 1 MEM**

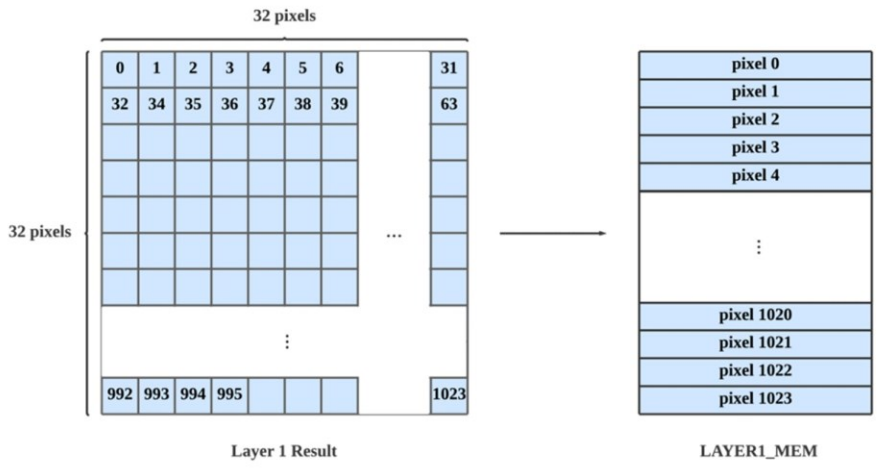


Figure 6. Layer 1 result memory mapping

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.

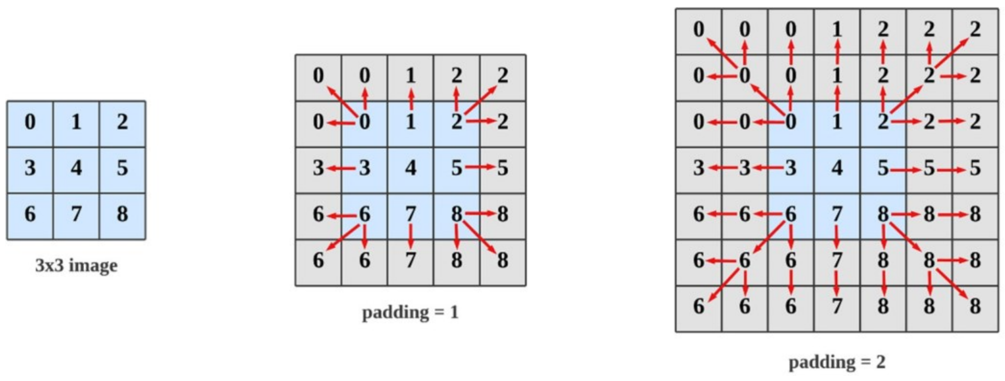


Figure 7. Replicate padding example

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.

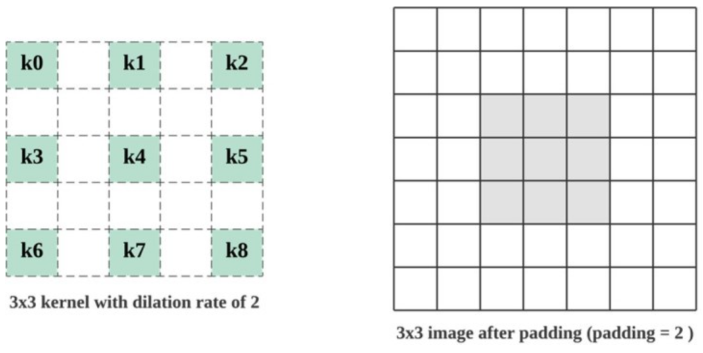
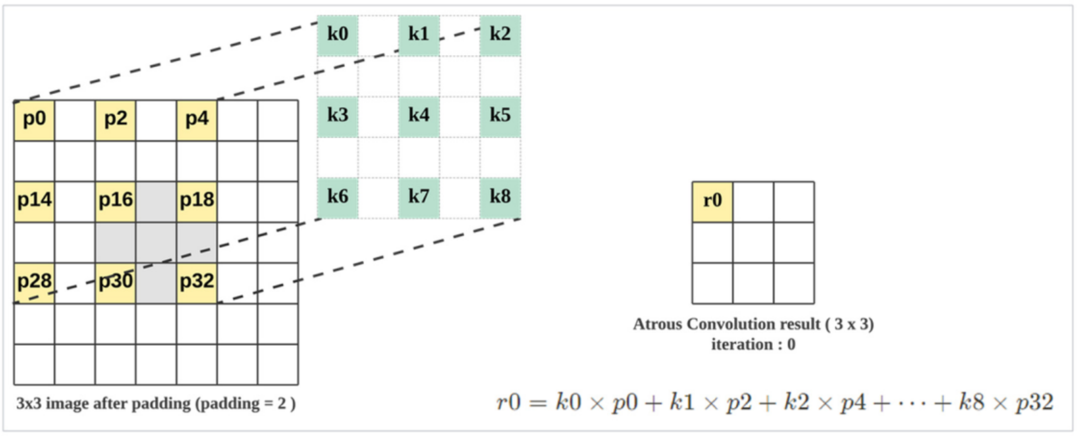
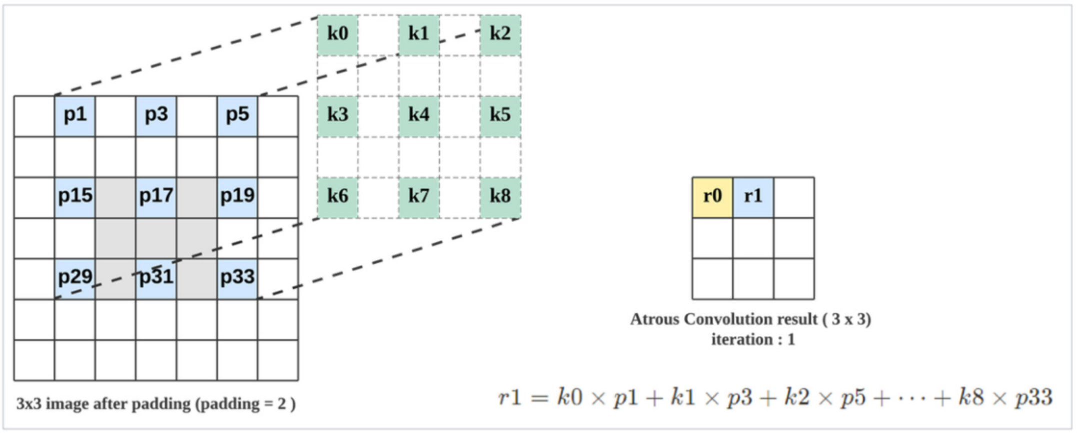

k0
k1
k2
k3
k4
k5
k6
k7
k8
3x3 kernel with dilation rate of 2
3x3 image after padding (padding = 2)

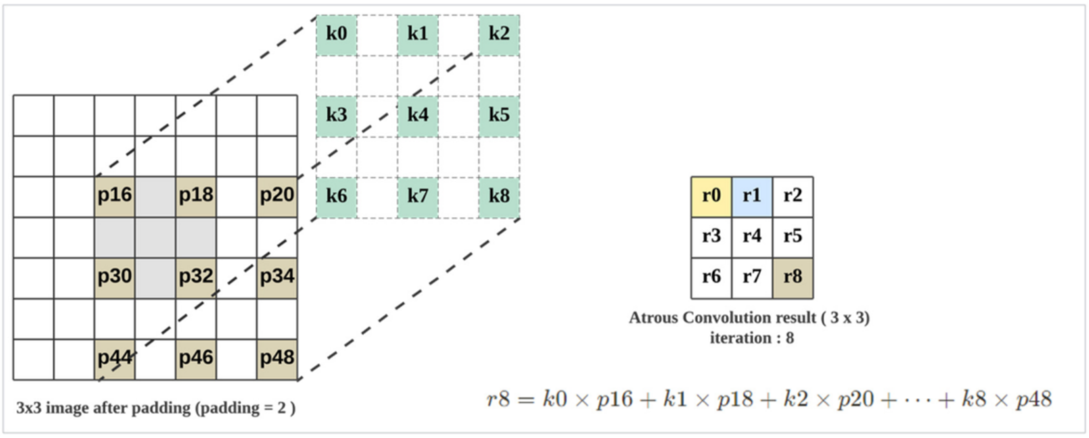
Figure 8. Kernel and image used in example


k0
k1
k2
k3
k4
k5
k6
k7
k8
p0
p2
p4
p14
p16
p18
p28
p30
p32
3x3 image after padding (padding = 2)
r0
Atrous Convolution result (3 x 3)
iteration : 0

$$r0 = k0 \times p0 + k1 \times p2 + k2 \times p4 + \cdots + k8 \times p32$$


k0
k1
k2
k3
k4
k5
k6
k7
k8
p1
p3
p5
p15
p17
p19
p29
p31
p33
3x3 image after padding (padding = 2)
r0
r1
Atrous Convolution result (3 x 3)
iteration : 1

$$r1 = k0 \times p1 + k1 \times p3 + k2 \times p5 + \cdots + k8 \times p33$$



$$r8 = k0 \times p16 + k1 \times p18 + k2 \times p20 + \cdots + k8 \times p48$$

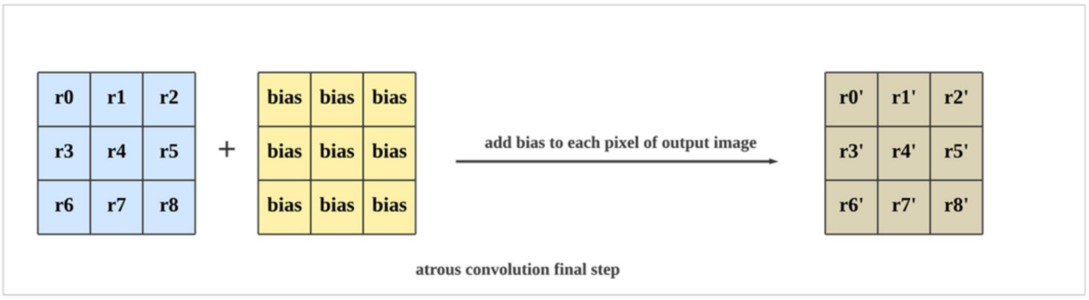
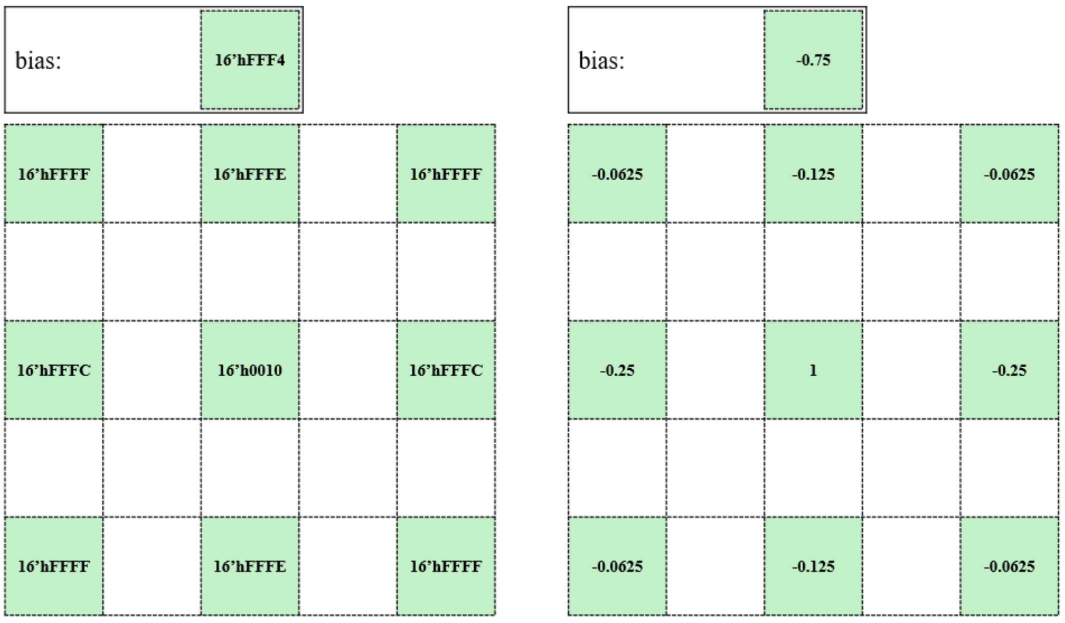

r0
r1
r2
r3
r4
r5
r6
r7
r8
+
bias
add bias to each pixel of output image
r0'
r1'
r2'
r3'
r4'
r5'
r6'
r7'
r8'
atrous convolution final step

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 12-bit integer part (MSB) and 4-bit fractional part (LSB), and the MSB represents sign bit.



Kernal and bias used in this assignment(hexadecimal)   Kernal and bias used in this assignment(decimal)

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.

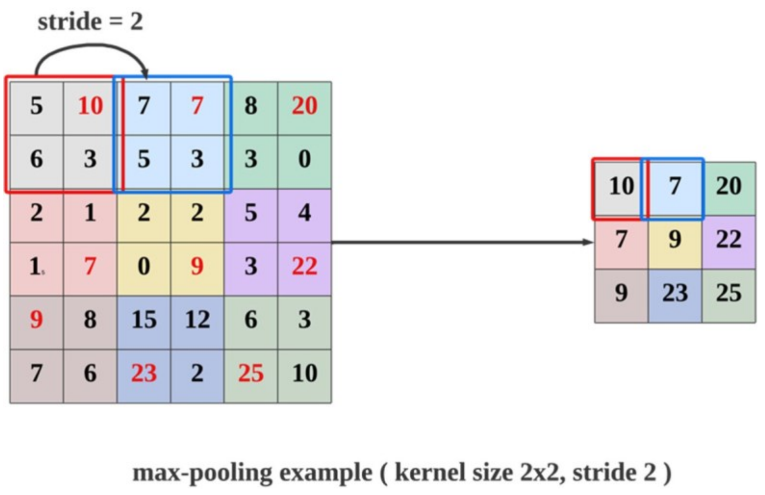


Figure 11. Max-pooling example

2.4. Timing Specifications:

2.4.1 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***.

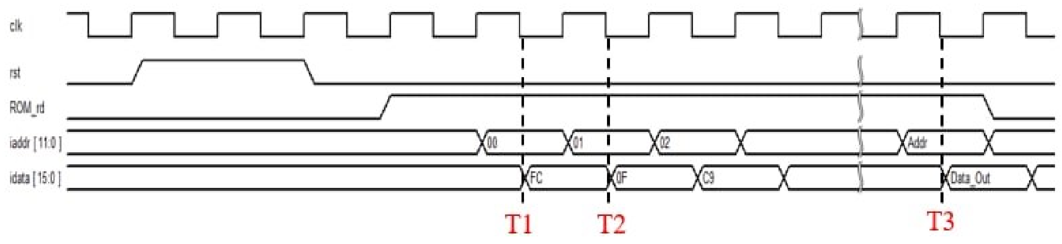


Figure 12. input image pixel retrieval control

2.4.2 Layer 0/1 MEM data reading control

When the ***layer0/1_web*** signal is set to 1, ATCONV circuit can read data from the Layer 0/1 memory. After assign the data address with ***layer0/1_A*** port at the positive edge of ***clk***, the testfixure would transmit the corresponding data to the ***layer0/1_Q*** port at the negative edge of clk. Therefore, ATCONV circuit can get the data at the next positive edge of ***clk***.

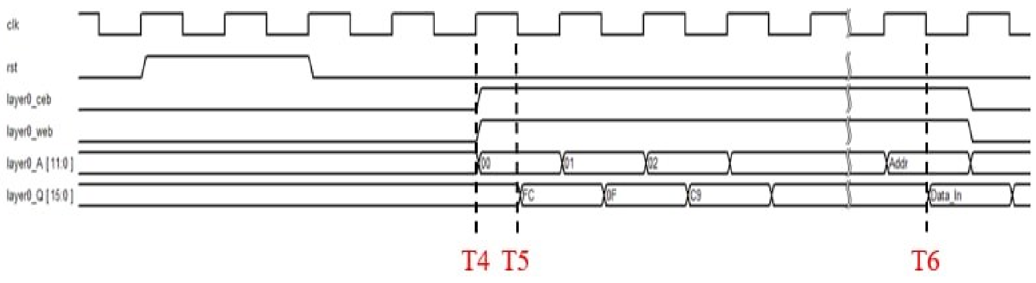


Figure 13. memory data reading control

2.4.3 Layer 0/1 MEM data writing control

When the ***layer0/1_web*** signal is set to 0, ATCONV circuit can write data into the Layer 0/1 memory. Assign the data address and the data to ***layer0/1_A*** and ***layer0/1_D*** 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***.

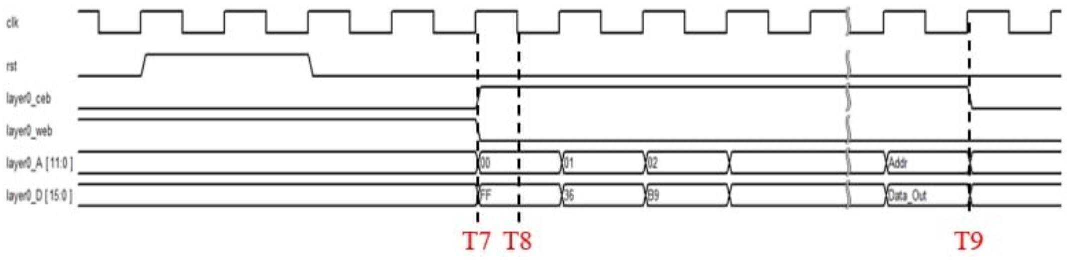


Figure 14. memory data writing control

3. System Design Specifications:

3.1 Block Overview:

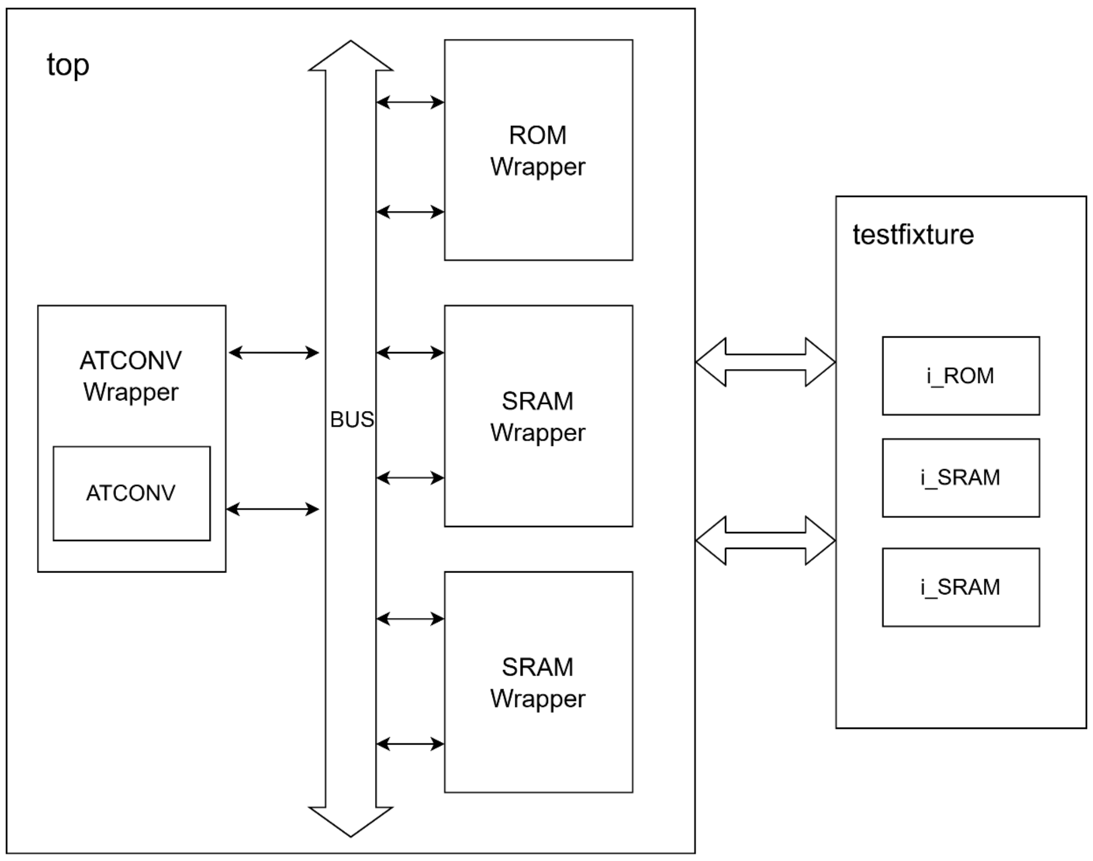


Figure 15. System block overview

3.2 I/O Interfaces & System Spec:

Signal Name	I/O	width	Description
top			
clk	I	1	System clock signal. This system is synchronized at the positive edge of the clock.
rst	I	1	Active-high asynchronous reset signal.
SRAM_D_L0	O	16	Layer 0 SRAM data in bus.
SRAM_A_L0	O	12	Layer 0 SRAM address bus.
SRAM_Q_L0	I	16	Layer 0 SRAM data out bus.
SRAM_ceb_L0	O	1	Layer 0 SRAM chip enable. Active high.
SRAM_web_L0	O	1	Layer 0 SRAM write enable. When this signal is 1, it indicates read SRAM; while 0, it indicates write SRAM.
SRAM_D_L1	O	16	Layer 1 SRAM data in bus.

SRAM_A_L1	O	12	Layer 1 SRAM address bus.
SRAM_Q_L1	I	16	Layer 1 SRAM data out bus.
SRAM_ceb_L1	O	1	Layer 1 SRAM chip enable. Active high.
SRAM_web_L1	O	1	Layer 1 SRAM write enable. When this signal is 1, it indicates read SRAM; while 0, it indicates write SRAM.
ROM_rd_IMG	O	1	ROM read enable signal.
ROM_A_IMG	O	12	ROM address bus.
ROM_Q_IMG	I	16	ROM data out bus.
done	O	1	When the system completes writing to layer 0/1 SRAM, it sets done to high to indicate completion.
ATCONV_Wrapper (Master 0)			
bus_clk	I	1	System clock signal. This system is synchronized at the positive edge of the clock.
bus_rst	I	1	Active-high asynchronous reset signal.
ID_M	O	2	Slave ID channel. See Table III below.
ADDR_M	O	12	Master address channel.
BLEN_M	O	4	Master burst length, range from 1 to 4.
RDATA_M	I	16	Master read data channel from slave.
RLAST_M	I	1	Master read last signal from slave. This signal indicates the last transfer in a read burst.
RVALID_M	O	1	Master read valid signal.
RREADY_M	I	1	Master read ready signal from slave.
WDATA_M	O	16	Master write data channel into slave.
WLAST_M	O	1	Master write last signal. This signal indicates the last transfer in a write burst.
WVALID_M	O	1	Master write valid signal.
WREADY_M	I	1	Master write ready signal from slave.
done	O	1	When the system completes writing to layer 0/1 SRAM, it sets done to high to indicate completion.
BUS (Bridge)			
bus_clk	I	1	System clock signal. This system is synchronized at the positive edge of the clock.
bus_rst	I	1	Active-high asynchronous reset signal.
ID_M0	I	2	Slave ID channel from Master 0.
ADDR_M0	I	12	Master 0 address channel.

BLEN_M0	I	4	Master 0 burst length, range from 1 to 4.
RDATA_M0	O	16	Master 0 read data channel.
RLAST_M0	O	1	Master 0 read last signal.
RVALID_M0	I	1	Master 0 read valid signal.
RREADY_M0	O	1	Master 0 read ready signal.
WDATA_M0	I	16	Master 0 write data channel.
WLAST_M0	I	1	Master 0 write last signal.
WVALID_M0	I	1	Master 0 write valid signal.
WREADY_M0	O	1	Master 0 write ready signal.
ADDR_S0	O	12	Slave 0 address channel.
BLEN_S0	O	4	Slave 0 burst length, range from 1 to 4.
RDATA_S0	I	16	Slave 0 read data channel.
RLAST_S0	I	1	Slave 0 read last signal to master.
RVALID_S0	O	1	Slave 0 read valid signal.
RREADY_S0	I	1	Slave 0 read ready signal.
ADDR_S1	O	12	Slave 1 address channel.
BLEN_S1	O	4	Slave 1 burst length, range from 1 to 4.
RDATA_S1	I	16	Slave 1 read data channel.
RLAST_S1	I	1	Slave 1 read last signal.
RVALID_S1	O	1	Slave 1 read valid signal.
RREADY_S1	I	1	Slave 1 read ready signal.
WDATA_S1	O	16	Slave 1 write data channel.
WLAST_S1	O	1	Slave 1 write last signal.
WVALID_S1	O	1	Slave 1 write valid signal.
WREADY_S1	I	1	Slave 1 write ready signal.
ADDR_S2	O	12	Slave 2 address channel.
BLEN_S2	O	4	Slave 2 burst length, range from 1 to 4.
RDATA_S2	I	16	Slave 2 read data channel.
RLAST_S2	I	1	Slave 2 read last signal.
RVALID_S2	O	1	Slave 2 read valid signal.
RREADY_S2	I	1	Slave 2 read ready signal.
WDATA_S2	O	16	Slave 2 write data channel.
WLAST_S2	O	1	Slave 2 write last signal.
WVALID_S2	O	1	Slave 2 write valid signal.
WREADY_S2	I	1	Slave 2 write ready signal.
SRAM_Wrapper (Slave 1/2)			

bus_clk	I	1	System clock signal. This system is synchronized at the positive edge of the clock.
bus_rst	I	1	Active-high asynchronous reset signal.
ADDR_S	I	12	Slave address channel.
BLEN_S	I	4	Slave burst length, range from 1 to 4.
RDATA_S	O	16	Slave read data channel to master.
RLAST_S	O	1	Slave read last signal to master. This signal indicates the last transfer in a read burst.
RVALID_S	I	1	Slave read valid signal from master.
RREADY_S	O	1	Slave read ready signal.
WDATA_S	I	16	Slave write data channel from master.
WLAST_S	I	1	Slave write last signal. This signal indicates the last transfer in a write burst.
WVALID_S	I	1	Slave write valid signal from master.
WREADY_S	O	1	Slave write ready signal.
SRAM_D	O	16	SRAM data in bus.
SRAM_A	O	12	SRAM address bus.
SRAM_Q	I	16	SRAM data out bus.
SRAM_ceb	O	1	SRAM chip enable. Active high.
SRAM_web	O	1	SRAM write enable. When this signal is 1, it indicates read SRAM; while 0, it indicates write SRAM.
ROM_Wrapper (Slave 0)			
bus_clk	I	1	System clock signal. This system is synchronized at the positive edge of the clock.
bus_rst	I	1	Active-high asynchronous reset signal.
ADDR_S	I	12	Slave address channel.
BLEN_S	I	4	Slave burst length, range from 1 to 4.
RDATA_S	O	16	Slave read data channel to master.
RLAST_S	O	1	Slave read last signal to master. This signal indicates the last transfer in a read burst.
RVALID_S	I	1	Slave read valid signal from master.
RREADY_S	O	1	Slave read ready signal.
ROM_rd	O	1	ROM read enable signal.
ROM_A	O	12	ROM address bus.
ROM_Q	I	16	ROM data out bus.

Table II. I/O interface of System

Module	Instance	ID
ROM_Wrapper	Slave_0	0
SRAM_Wrapper	Slave_1	1
SRAM_Wrapper	Slave_2	2
Default Slave		3

Table III. Slave ID number and instance name

3.3 Bus Interface Specification

◆ **Handshake**

In this assignment, system read/write operation should be done via AXI-liked bus interface while valid/ready handshake is the most vital action to activate the data transaction. Therefore, there are two types of handshake process in this assignment which is read channel handshake and write channel handshake. In this assignment, for simplicity, ready signal depends on valid signal which implies that slave interface is controlled by master interface. Namely, the valid signal in this assignment should always be pulled high before ready is set to 1, below figure will show the behavior of the handshake process.

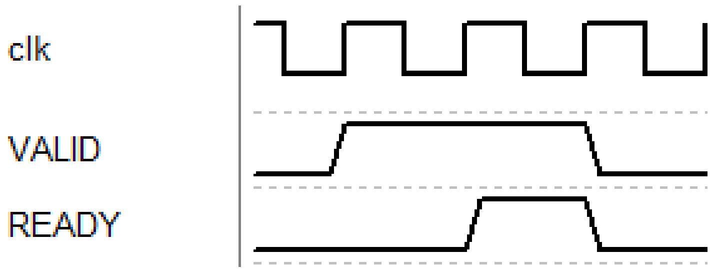


Figure 16. Handshake process

◆ Read Transaction

In this assignment, master interface should be able to read data from different slave interfaces. While system operating read transaction, address, slave ID and burst length settings should be prepare before read channel handshake, these data will then be sent from master to slave interface via bus interface. While the amount of data read from slave interface is equivalent to the burst length, read last signal should be pull high immediately.

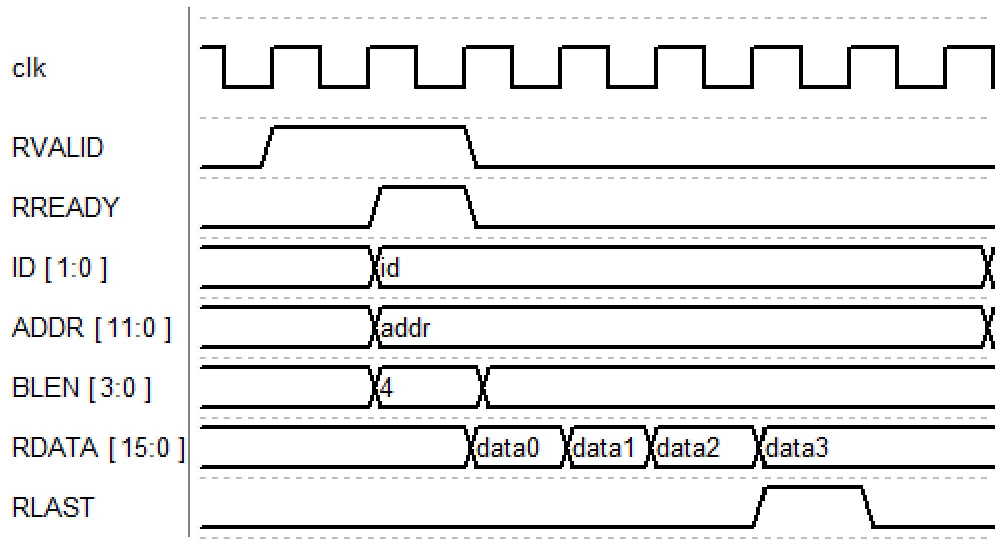


Figure 17. Read transaction

◆ Write Transaction

In this assignment, master interface should be able to write data into different slave interfaces. While system operating write transaction, address, slave ID and burst length settings should be prepare before write channel handshake, these data will then be sent from master to slave interface via bus interface. While the amount of data written into slave interface is equivalent to the burst length, write last signal should be pull high immediately.

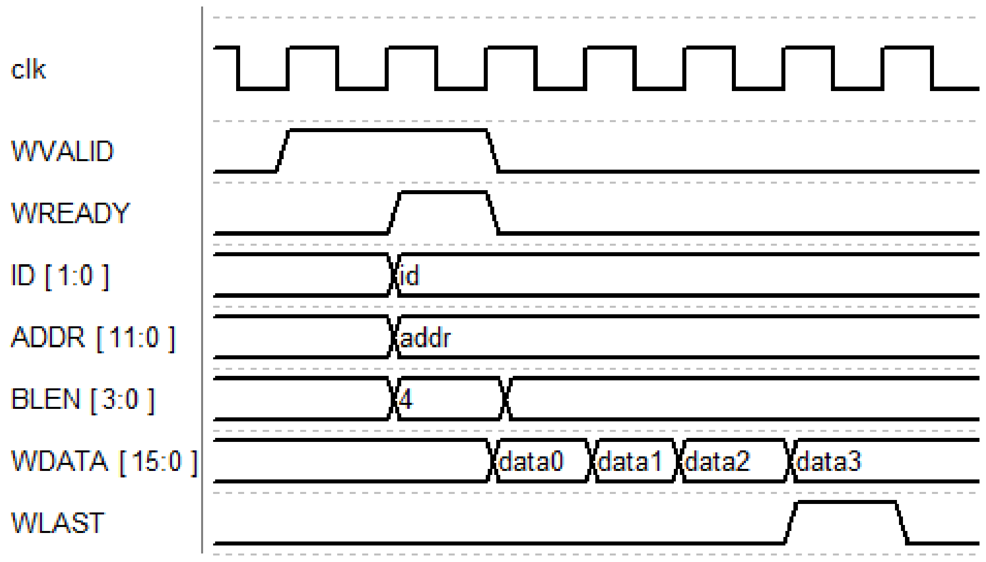


Figure 18. Write transaction

4. File Description:

File Name	Description
file/ATCONV/	
ATCONV.v	The top module of ATCONV.
testfixture.sv	The testbench file. You can only change the pattern number and END_CYCLE.
file/ATCONV/dat/tb1~tb3/	
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.
file/ATCONV/mem/	
SRAM.sv	Static random-access memory for ATCONV.
ROM.sv	Read-only memory for ATCONV.
file/System/	
ATCONV_Wrapper.v	System master interface of ATCONV.
BUS.v	System bus interface.
top.v	Top module of the entire system. You are not allowed to modify this file.
testfixture.sv	The testbench file. You can only change the pattern number and END_CYCLE.
file/System/dat/tb1~tb3/	
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.
file/System/include/	
define.v	Parameter definition file.
file/System/mem/	
SRAM_Wrapper.v	System slave interface of layer 0/1 SRAM.
ROM_Wrapper.v	System slave interface of image ROM.
SRAM.sv	Static random-access memory instance.
ROM.sv	Read-only memory instance.

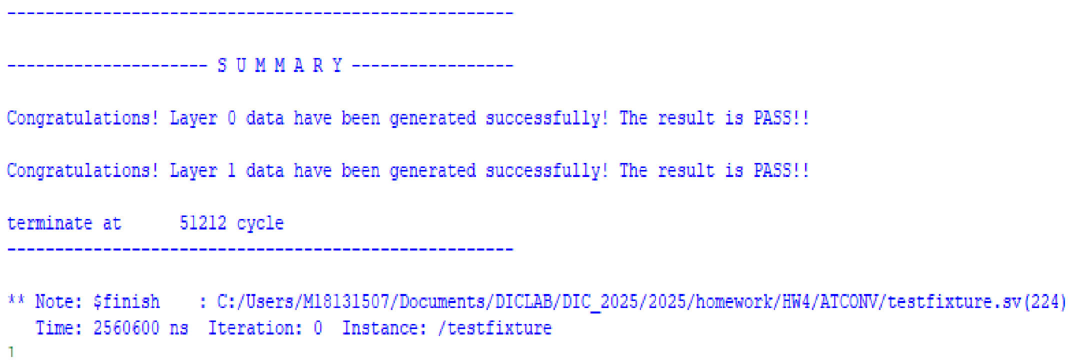
Table IV. File structure and description

5. Scoring:

5.1 ATCONV [60%]

- **Functional Simulation [40%]**

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 20% of the score each.



```
-----------------------------------------------------

-------------------- 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      51212 cycle
-----------------------------------------------------

** Note: $finish    : C:/Users/M18131507/Documents/DICLAB/DIC_2025/2025/homework/HW4/ATCONV/testfixture.sv(224)
   Time: 2560600 ns  Iteration: 0  Instance: /testfixture
1
```

Figure 19. Functional simulation result example

◆ **Pre-Layout Simulation [20%]**

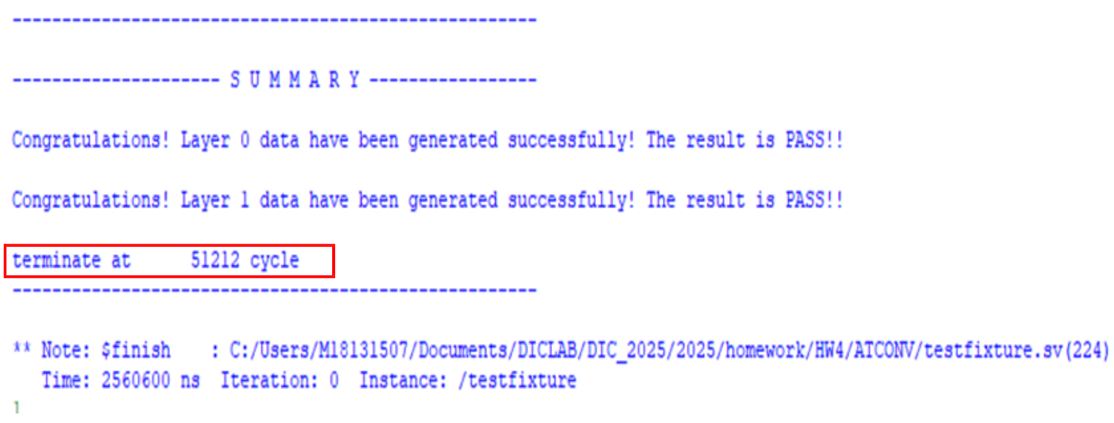
✓ **Synthesis:**

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

DEVICE ：Cyclone IV E - EP4CE55F23A7

✓ **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 pre-layout simulation results of Layer 0 and Layer 1 take 10% of the score each.



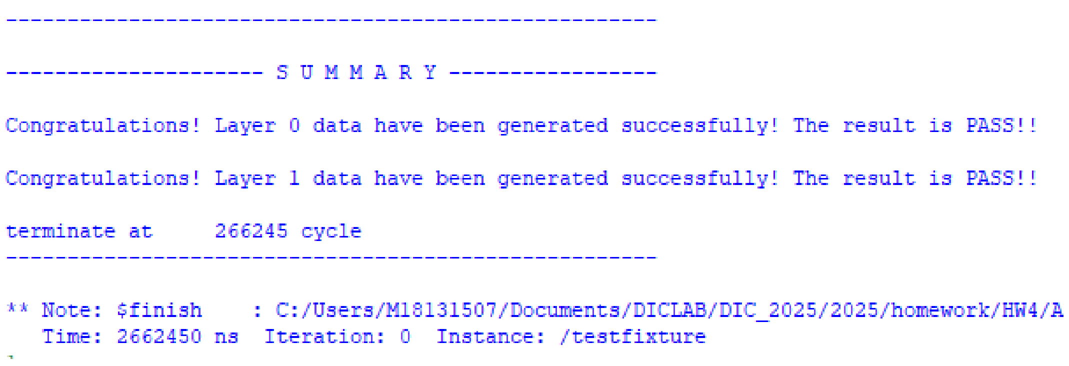
```
-----------------------------------------------------

-------------------- 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       51212 cycle
-----------------------------------------------------

** Note: $finish    : C:/Users/M18131507/Documents/DICLAB/DIC_2025/2025/homework/HW4/ATCONV/testfixture.sv(224)
   Time: 2560600 ns  Iteration: 0  Instance: /testfixture
```

Figure 20. Pre-layout simulation result example

5.2 System [30%]

◆ **Functional Simulation [20%]**

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 10% of the score each.



```
-----------------------------------------------------

-------------------- 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      266245 cycle
-----------------------------------------------------

** Note: $finish    : C:/Users/M18131507/Documents/DICLAB/DIC_2025/2025/homework/HW4/A
   Time: 2662450 ns  Iteration: 0  Instance: /testfixture
```

Figure 21. Functional simulation result example

◆ **Synthesis**

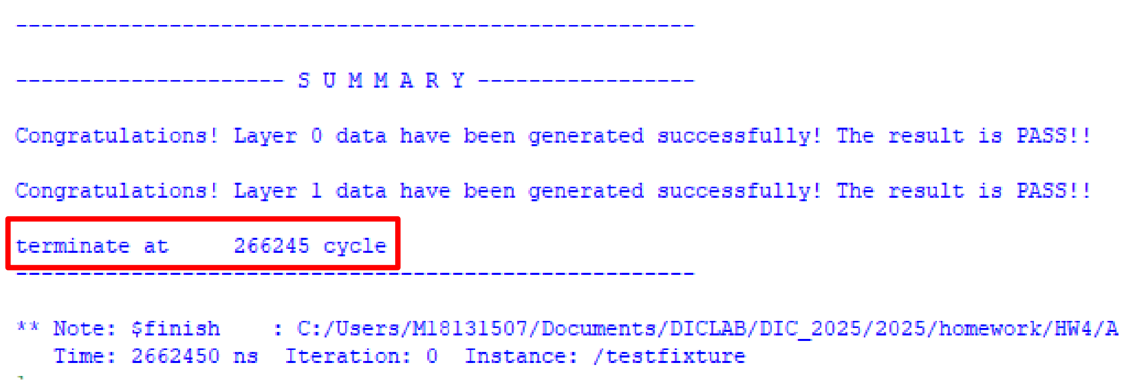
Your code should be synthesizable. After it is synthesized in Quartus, a file named top.vo will be obtained.

DEVICE : Cyclone IV E - EP4CE55F23A7

◆ **Pre-Layout Simulation [10%]**

All of the results should be generated correctly using top.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 pre-layout simulation results of Layer 0 and Layer 1 take 5% of the score each.

All of the result should be generated correctly, and you will get the following message in ModelSim simulation.



```
-----------------------------------------------------

--------------------- 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      266245 cycle
-----------------------------------------------------

** Note: $finish    : C:/Users/M18131507/Documents/DICLAB/DIC_2025/2025/homework/HW4/A
   Time: 2662450 ns  Iteration: 0  Instance: /testfixture
```

Figure 22. Pre-layout simulation result example

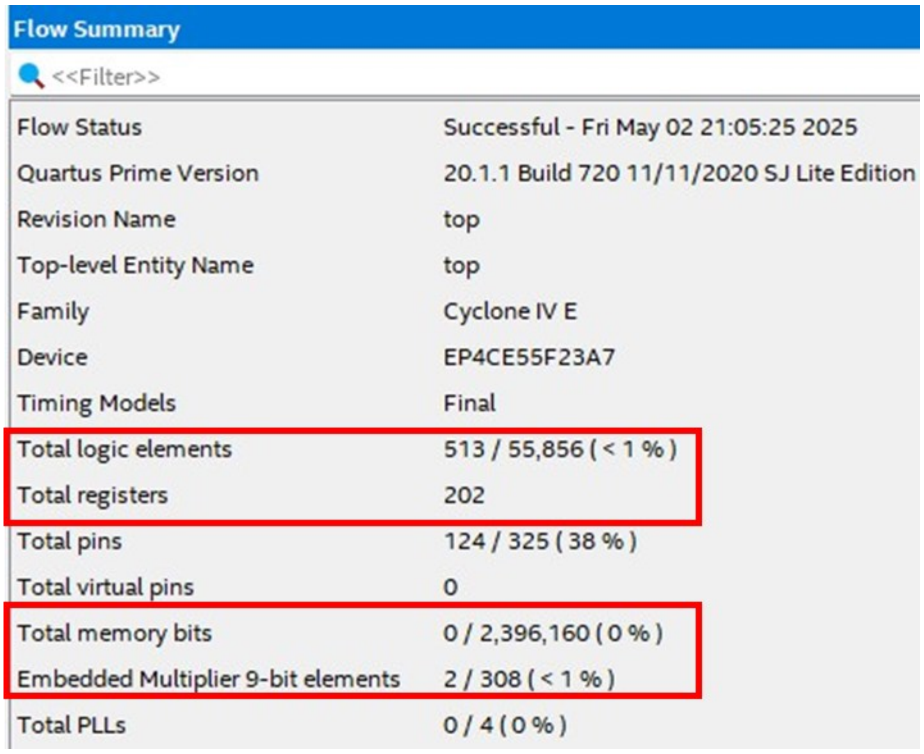
5.3 Performance [10%]

The performance is scored by the total logic elements, total registers, 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 System Pre-Layout 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 register + Total memory bits + 9*Embedded multiplier 9-bit elements) × (Total cycle)

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



Flow Summary

<<Filter>>

Flow Status	Successful - Fri May 02 21:05:25 2025
Quartus Prime Version	20.1.1 Build 720 11/11/2020 SJ Lite Edition
Revision Name	top
Top-level Entity Name	top
Family	Cyclone IV E
Device	EP4CE55F23A7
Timing Models	Final
Total logic elements	513 / 55,856 (< 1 %)
Total registers	202
Total pins	124 / 325 (38 %)
Total virtual pins	0
Total memory bits	0 / 2,396,160 (0 %)
Embedded Multiplier 9-bit elements	2 / 308 (< 1 %)
Total PLLs	0 / 4 (0 %)

Figure 23. Quartus synthesis result example

6. Submission:

6.1 Submitted files

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

StudentID/	
ATCONV/	
	RTL category/
ATCONV.v	All of your Verilog RTL code
	Pre-Layout category/
ATCONV.vo	Pre-Layout netlist generated by Quartus
ATCONV_v.sdo	Pre-Layout netlist generated by Quartus
System/	
	RTL category/
ATCONV_Wrapper.v	ATCONV master interface.
SRAM_Wrapper.v	SRAM slave interface.
ROM_Wrapper.v	ROM slave interface.
BUS.v	Bus interface.
	Pre-Layout category/
top.vo	Pre-Layout netlist generated by Quartus
top_v.sdo	Pre-Layout netlist generated by Quartus
Documentary/	
Report.pdf	The report file of your design (in pdf).

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. 23) of your synthesized design. And fill the field of total cycle used according to the pre-layout simulation results that ModelSim shows. If the filled-in values don't match your synthesized design or pre-layout simulation results, you will lose 10 points.

6.3 Note

- In this homework, the clock width is fixed at 50 ns, so you are not allowed to modify "CYCLE" definition in testbench file. You are only allowed to modify the defined “END_CYCLE” and “tb1/tb2/tb3” in testbench file. “END_CYCLE” decides the maximum cycles your circuit takes to complete the simulation; and “tb1/tb2/tb3” decides the test pattern to be tested. Please do not modify any other content of the testbench.

- Please submit your .zip file to folder HW4 Submission in moodle. Deadline: 2025/06/01 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.
- Late submissions will result in a penalty of 5 points per day.