2021 Digital IC Design

Homework 4: Median Filter Engine

1 Introduction

The median filter is a non-linear digital filtering technique, which is often used to remove noise from an image or signal. Such noise reduction is a typical pre-processing step to improve the results of later processing (for example, edge detection on an image). In this homework, please implement an median filter engine (MFE), which uses median filter to convolve the image. The specification and function of MFE circuit will be described in detail in the following section.

2 Design Specifications

2.1 Block overview

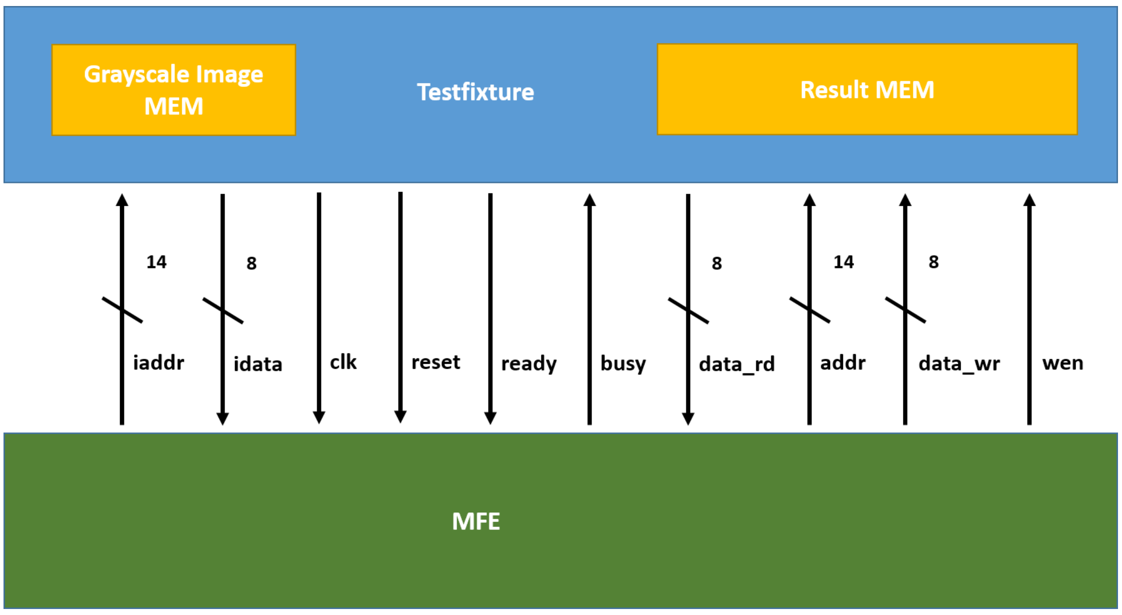


Fig. 1 – System operation flow

2.2 I/O Interface

Name	I/O	Width	Description
clk	I	1	System clock signal. This system is synchronized with the positive edge of the clock.
reset	I	1	Active-high asynchronous reset signal.
ready	I	1	Grayscale image ready indication signal. When this

			signal is high, the grayscale image memory is already prepared. MFE can start sending grayscale image address to obtain the data.
busy	O	1	System busy indication signal. When MFE receives high level ready signal and MFE is ready to start the calculation, this signal has to be set as high. After the calculation is completed and the result is completely output, set this signal to low and the testbench will start checking if the result is correct.
iaddr	O	14	The signal for grayscale image memory address, which is used to request grayscale image pixel date.
idata	I	8	Input grayscale image pixel data signal, which represents an 8 bits unsigned integer. The testbench will send the pixel data according to the address *iaddr* indicates.
data_rd	I	8	Result memory read data signal, which represents an 8 bits unsigned integer. This signal is used to send the result memory data to MFE circuit.
addr	O	14	Result memory read/write address. This signal indicates which address of the result memory will be read or written.
data_wr	O	8	Result memory write data signal, which represents an 8 bits unsigned integer. The operation result of the MFE circuit is written to result memory using this signal.
wen	O	1	Result memory write enable signal. When this signal is low (read), the data with address *addr* in result memory is sent by *data_rd*. When this signal is high (write), the data sent by data_wr is written to the address *addr* in result memory.

2.3 Function Description

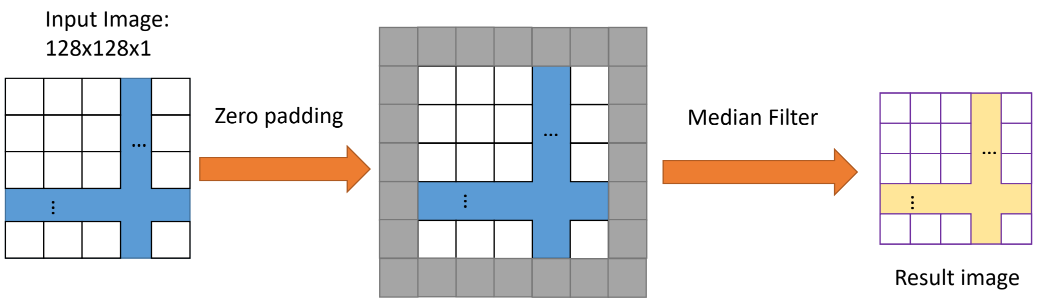


Fig. 2 – Circuit operation flow

The size of the input image in this system is 128x128, which is stored in the grayscale image memory. The correspondence between each pixel of the grayscale image and memory arrangement is shown in Figure 4. In the operation process, the MFE circuit uses the *iaddr* signal to send the address of requested image data (as shown in Figure 3. t1 time point). After the negative edge of each clock, the pixel data at requested address will be sent into the MFE circuit by the *idata* signal (as shown in figure 3. t2 time point).

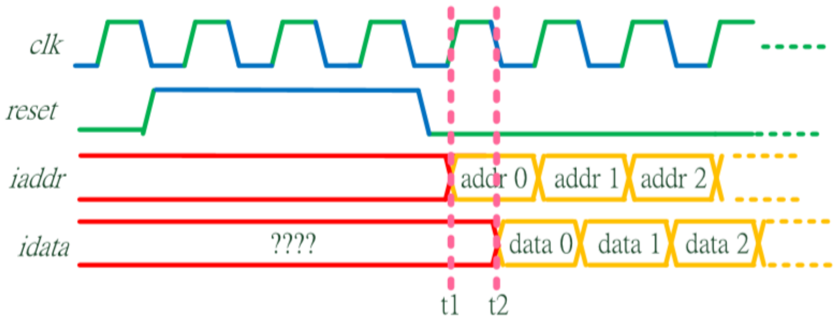


Fig. 3 – Timing diagram while reading grayscale image memory

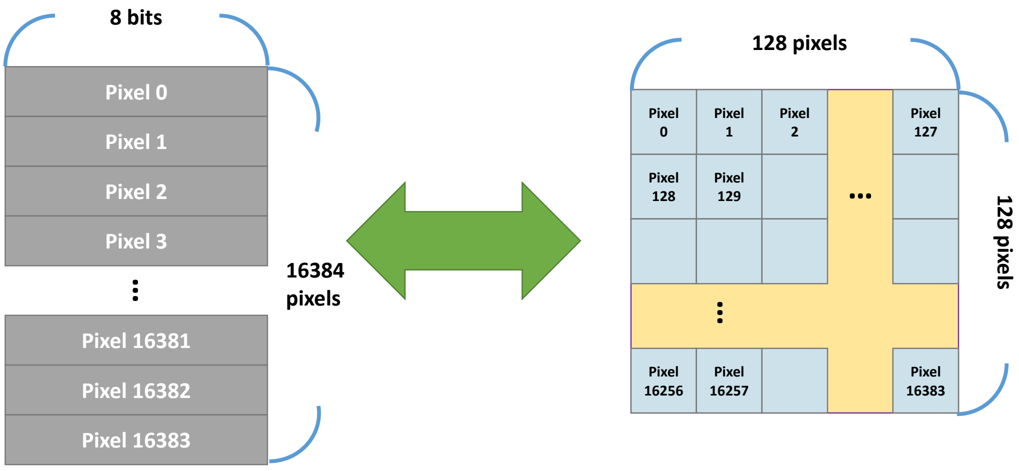


Fig. 4 – Arrangement mapping of input grayscale image and grayscale memory

The first step of the system is to perform zero-padding on the image to avoid size-changing of the image when doing MFE operation. Then the padded image is convolved with median filter to obtain result image. After the calculation is finished, the busy signal should be set to 0, and then testbench will start the verification process.

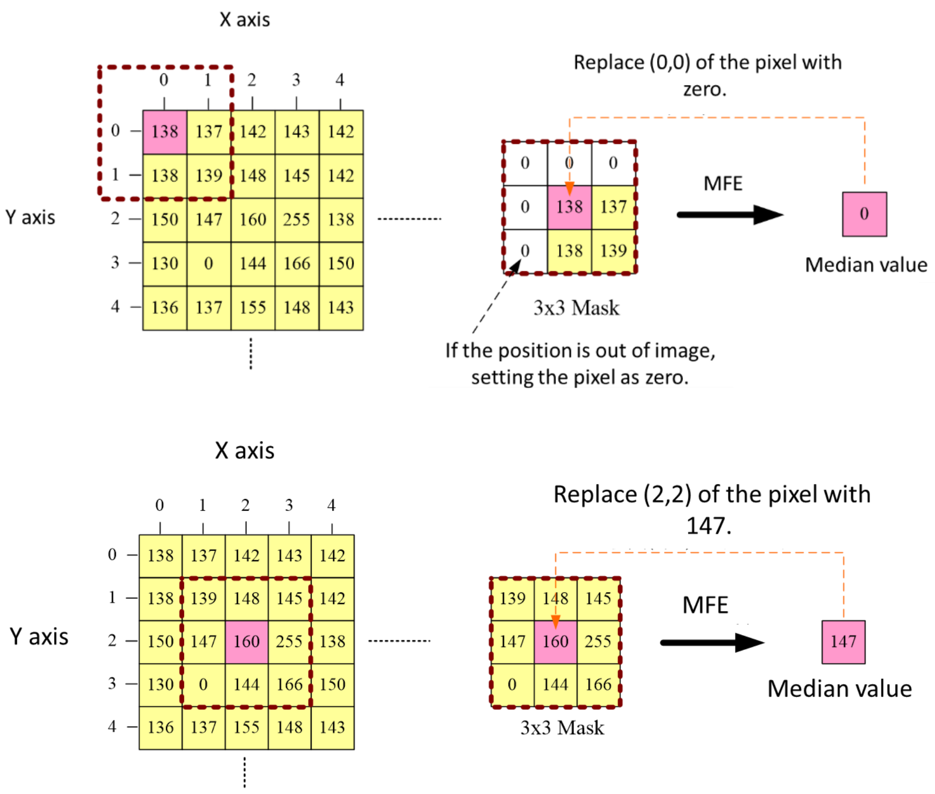

X axis
Y axis
Replace (0,0) of the pixel with zero.
MFE
Median value
3x3 Mask
If the position is out of image, setting the pixel as zero.
Replace (2,2) of the pixel with 147.
MFE
Median value
3x3 Mask

Fig. 5 – The median filtering operation.

In the reading phase, the *addr* signal represents the memory address of result memory to read and the *data_rd* signal will provide the data. *wen* signal should be set as low. The timing diagram of result memory reading operation is shown in figure 6.

In the writing phase, the *addr* signal informs the memory address to be written and the *data_wr* signal should provide the writing data. *wen* signal should be set as high. The timing diagram of result memory writing operation is shown in figure 7.

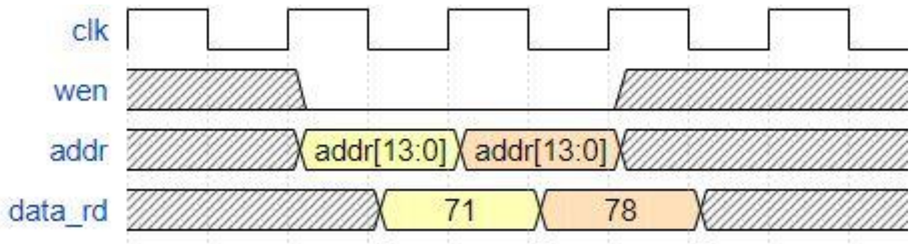


Fig. 6 – Timing diagram of reading result memory.

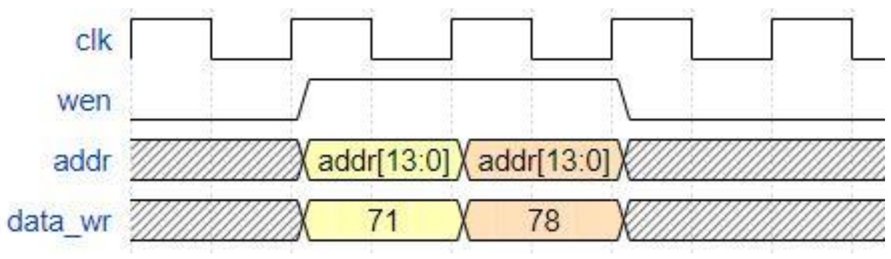


Fig. 7 – Timing diagram of writing result memory.

3 Software Verification

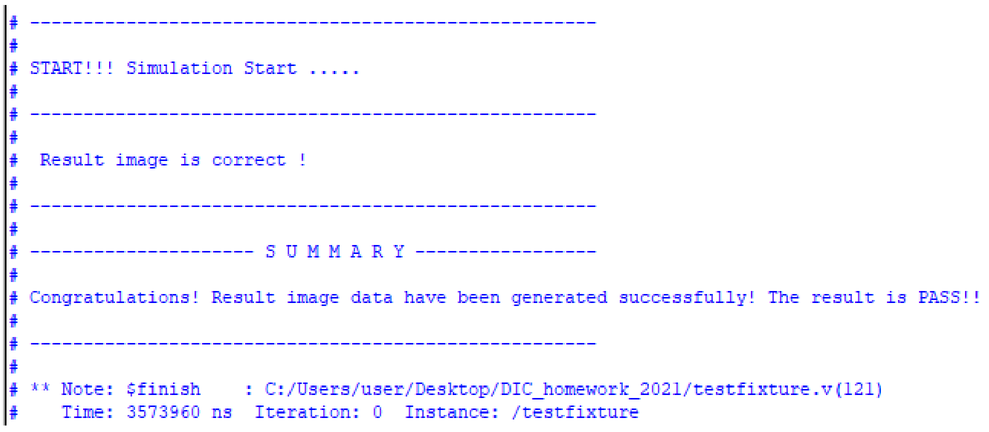
You need to write a program to verify your circuit. You can write your program by using C, C++ and python. The requirements and functionality of this program should be as following:

1. Be able to read any jpg image. The file path for reading has to be ./image.jpg.
2. Convert RGB image to gray scale.
3. Resize the image into 128x128.
4. Add salt and pepper noise to the image.
   (You may set the probability of the noise to 0.02.)
5. Process median filtering to the image with noise.
6. Output the image with noise and the filtered image as img.dat and golden.dat, respectively. The format of the output files should be the same as the img.dat and golden.dat which TA gave. The output file should be able to be read by the testbench and simulate correctly.
7. Please name your program in the format: main.c, main.cpp or main.py.

4 Scoring

4.1 Functional Simulation **[40%]**

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



```
# -----------------------------------------------------
#
# START!!! Simulation Start .....
#
# -----------------------------------------------------
#
#  Result image is correct !
#
# -----------------------------------------------------
#
# -------------------- S U M M A R Y ----------------
#
# Congratulations! Result image data have been generated successfully! The result is PASS!!
#
# -----------------------------------------------------
#
# ** Note: $finish    : C:/Users/user/Desktop/DIC_homework_2021/testfixture.v(121)
#    Time: 3573960 ns  Iteration: 0  Instance: /testfixture
```

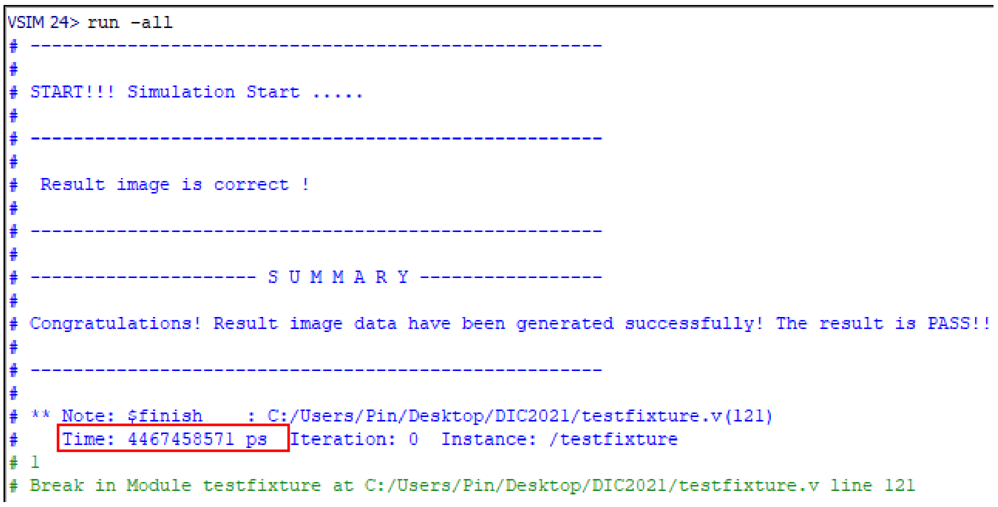
4.2 Gate Level Simulation **[20%]**

4.2.1 Synthesis

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

4.2.2 Simulation

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



```
VSIM 24> run -all
# ----------------------------------------------------
#
# START!!! Simulation Start .....
#
# ----------------------------------------------------
#
#  Result image is correct !
#
# ----------------------------------------------------
#
# -------------------- S U M M A R Y ----------------
#
# Congratulations! Result image data have been generated successfully! The result is PASS!!
#
# ----------------------------------------------------
#
# ** Note: $finish    : C:/Users/Pin/Desktop/DIC2021/testfixture.v(121)
#    Time: 4467458571 ps  Iteration: 0  Instance: /testfixture
# 1
# Break in Module testfixture at C:/Users/Pin/Desktop/DIC2021/testfixture.v line 121
```

4.3 Performance **[20%]**

The performance is scored by the total logic elements, total memory bit, and embedded multiplier 9-bit element your design used in gate-level simulation and the simulation time your design takes. The score will be decided by your ranking in all received homework. (The smaller, the better)

*Scoring = (Total logic elements + total memory bit + 9*embedded multiplier 9-bit element) × (longest gate-level simulation time in ns)*

4.4 Software Verification **[20%]**

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 the testing data which TA provided and the testing data generated by your software program.

5 Submission

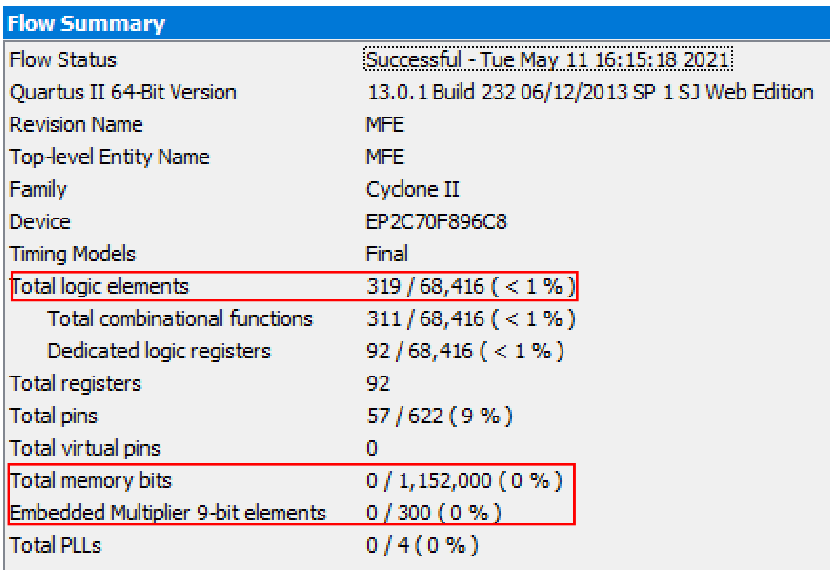
5.1 Submitted files

You should classify your files into four 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	SDF timing information generated by Quartus
	Documentary category
*.pdf	The report file of your design (in pdf).
	Verification category
*	The verification program.

5.2 Report file

Please follow the spec of report. If your report does not meet the spec, you may lose part of score. 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, and embedded multiplier 9-bit elements according to the flow summary of your synthesized design. And fill the field of gate-level simulation time according to the gate-level simulation result that Modelsim shows.



Flow Summary	
Flow Status	Successful - Tue May 11 16:15:18 2021
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
Revision Name	MFE
Top-level Entity Name	MFE
Family	Cyclone II
Device	EP2C70F896C8
Timing Models	Final
Total logic elements	319 / 68,416 (< 1 %)
Total combinational functions	311 / 68,416 (< 1 %)
Dedicated logic registers	92 / 68,416 (< 1 %)
Total registers	92
Total pins	57 / 622 (9 %)
Total virtual pins	0
Total memory bits	0 / 1,152,000 (0 %)
Embedded Multiplier 9-bit elements	0 / 300 (0 %)
Total PLLs	0 / 4 (0 %)

5.3 Note

In this homework, you are allowed to modify the defined CYCLE in testbench file. End_CYCLE, which decides the maximum cycles your circuit took to complete simulation, can also be modified according to your design. Please do not modify any other content of testbench.

Please submit your .zip file to folder HW4 in moodle.
Deadline: 2021/6/14 23:55
If you have any problem, please contact TA by email.
hcc7391@gmail.com
lt2es.93039@gmail.com