2023 Digital IC Design

Homework 5: Image Demosaicing

1. Introduction

Image demosaicing is the process of reconstructing a full-color image from a single sensor that captures only a portion of the color information at each pixel location. The Bayer pattern is a common type of color filter array used in digital cameras and smartphones. It uses a pattern of red, green, and blue filters arranged in a checkerboard-like pattern. In this homework, you are requested to design a circuit to transform an image in the Bayer pattern to a full-color image.

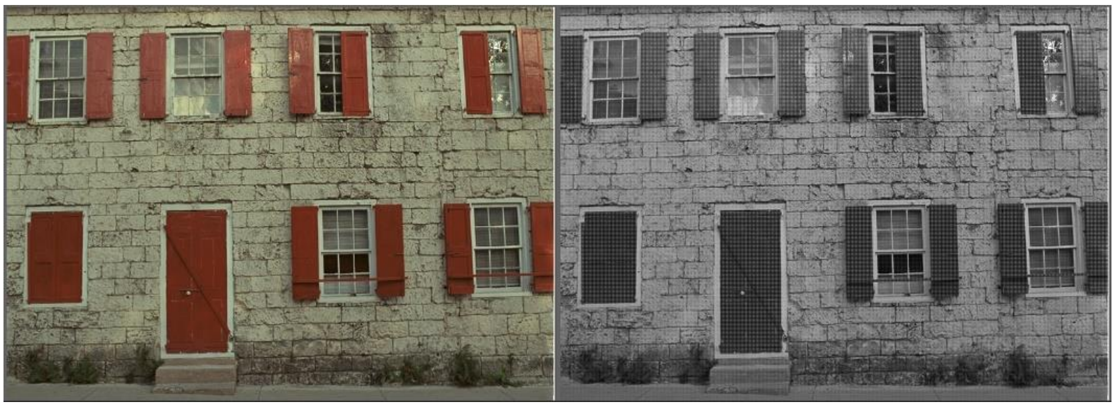


Figure 1. (a)Full color image (b)Bayer pattern image

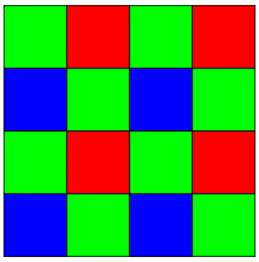


Figure 2. Bayer pattern

A basic demosaicing technique is bilinear interpolation, which reconstructs missing color values by averaging the values of neighboring pixels. Bilinear interpolation will serve as the baseline algorithm in this homework. However, you are encouraged to design more advanced algorithms to achieve higher reconstruction quality. The quality of the reconstructed images will be evaluated using the Peak Signal-to-Noise Ratio (PSNR), which measures the difference between the original full-color image and the reconstructed image based on their respective signal strengths. More detailed circuit functionalities will be described in the subsequent sections.

2. Design Specifications

2.1 Block Overview

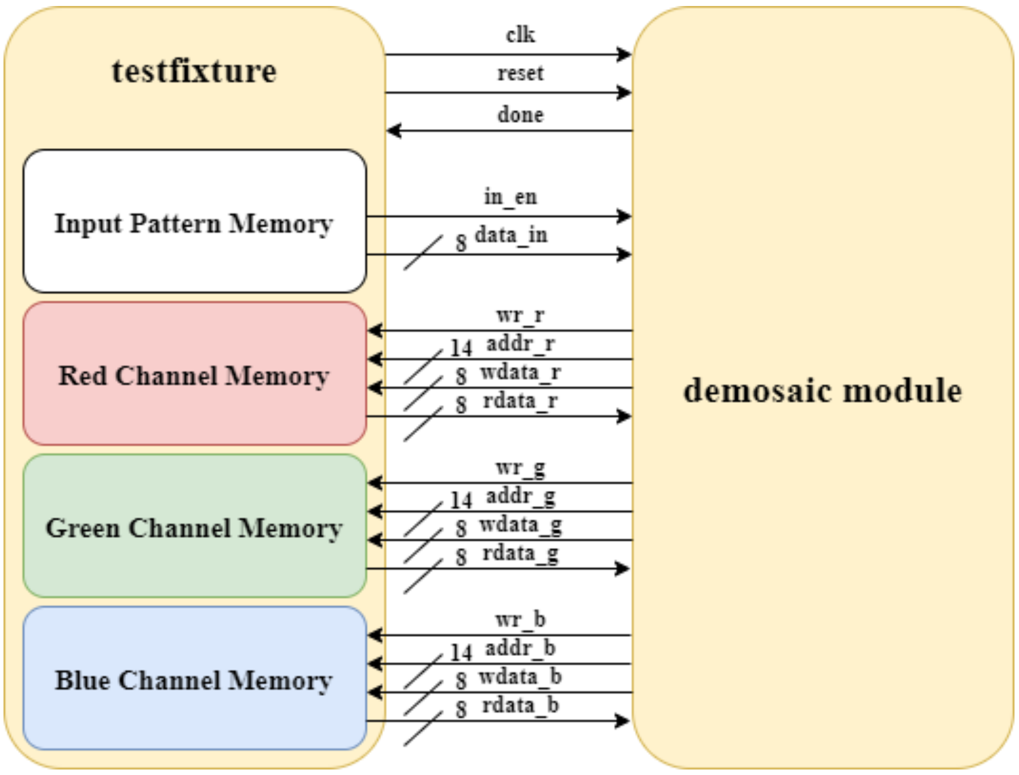


Figure 3. System Block Diagram

2.2 I/O Interface

Table I. I/O interface of the design

Signal Name	I/O	width	Description
clk	I	1	This circuit is a synchronous design triggered at the positive edge of *clk*.
reset	I	1	Active-high asynchronous reset signal.
in_en	I	1	Iuput data valid signal. When this signal is high, the value of *data_in* is valid.
data_in	I	8	8-bit unsigned Bayer pattern input data.
wr_r	O	1	Write/Read enable signal of the Red channel memory. When this signal is high, the value of *wdata_r* will be written to memory. When this signal is low, the value of *addr_r* in memory will

			be read to *rdata_r*.
addr_r	O	14	Write/Read address signal of the Red channel memory.
wdata_r	O	8	Write data signal of the Red channel memory.
rdata_r	I	8	Read data signal of the Red channel memory.
wr_g	O	1	Write/Read enable signal of the Green channel memory. When this signal is high, the value of *wdata_g* will be written to memory. When this signal is low, the value of *addr_g* in memory will be read to *rdata_g*.
addr_g	O	14	Write/Read address signal of the Green channel memory.
wdata_g	O	8	Write data signal of the Green channel memory.
rdata_g	I	8	Read data signal of the Green channel memory.
wr_b	O	1	Write/Read enable signal of the Blue channel memory. When this signal is high, the value of *wdata_b* will be written to memory. When this signal is low, the value of *addr_b* in memory will be read to *rdata_b*.
addr_b	O	14	Write/Read address signal of the Blue channel memory.
wdata_b	O	8	Write data signal of the Blue channel memory.
rdata_b	I	8	Read data signal of the Blue channel memory.
done	O	1	After finishing the demosaicing algorithm, set this signal to high. The testfixture will write the values stored in memories to a raw image.

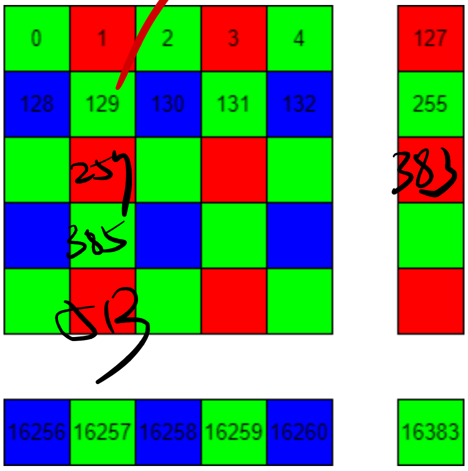
2.3 Function Description

2.3.1 Bayer Pattern Sequence

The input bayer pattern has a size of 128 x 128 pixels, so there are total 16384 pixels. Fig. 4 shows the input Bayer pattern sequence. Each pixel of the Bayer pattern has only one color channel. The odd rows of the Bayer pattern are green and red interlaced, and its even rows are blue and green interlaced. The demosaicing process reconstructs the missing color channels to obtain a full-color image. The pixels will be inputted from left to right, and from top to bottom. The memory address of each pixel is also arranged in the same manner.

起[illegible]

126×126
= 15876

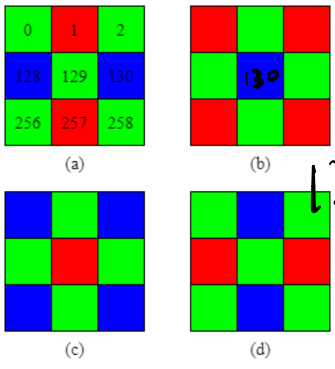


129 130 ... 254
257 258 ......

Figure 4. Input Bayer pattern sequence

2.3.2 Bilinear Interpolation

Bilinear interpolation is a common method used in image processing to estimate missing pixel values based on neighboring pixels. For demosaicing, it is used to estimate values of missing color channels by averaging the values of the required color channels of the neighboring pixels. The interpolation process can be divided into four cases based on the color of the pixel to be interpolated and its surrounding pixels, which is showed in Fig. 5.



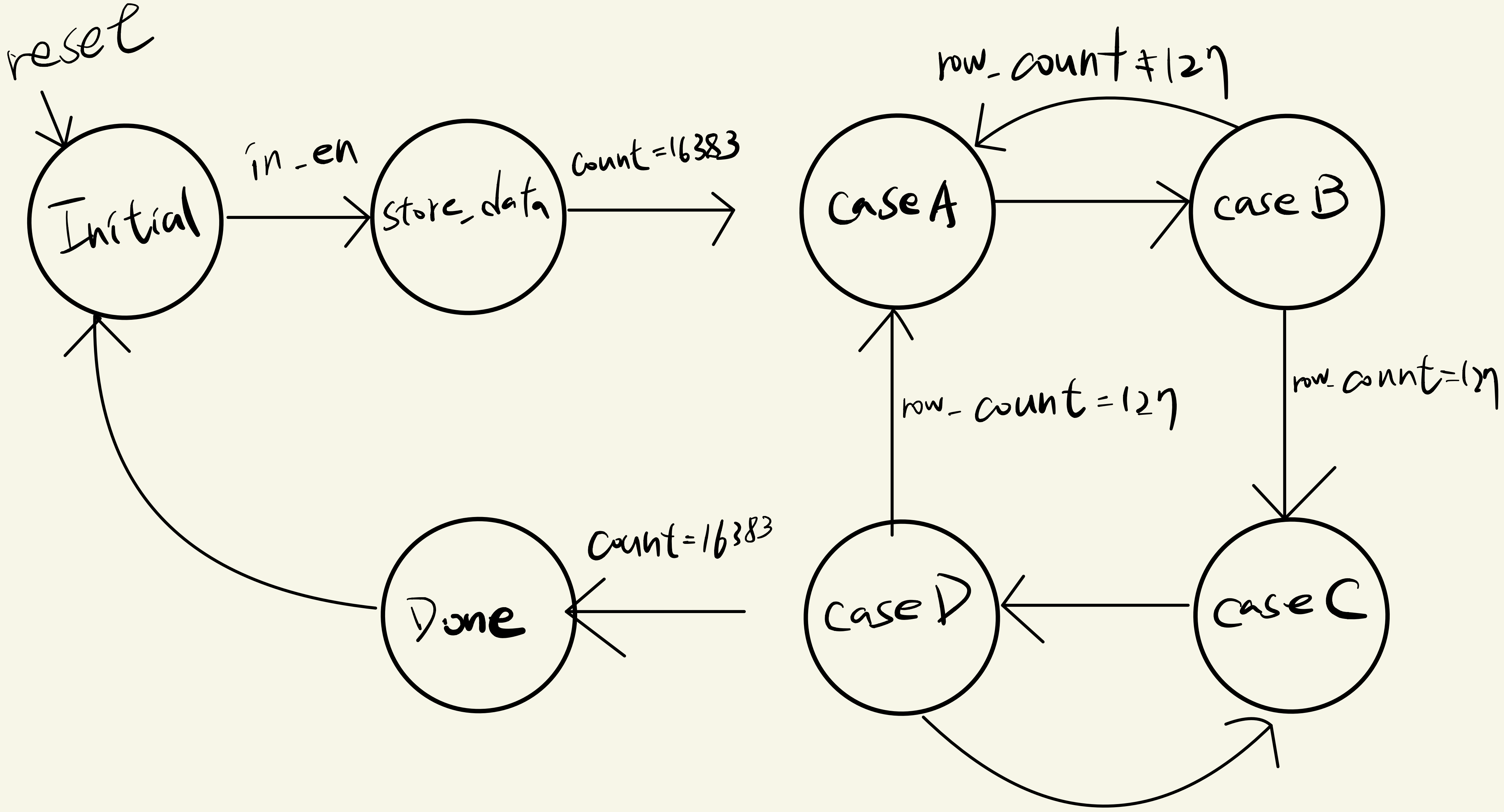
126
abab ... ab
126 cd cd ... cd
⋮
abab ... ab
cd cd ... cd

Figure 5. Four cases for bilinear interpolation (a)missing B, R on G (b)missing G, R on B (c)missing G, B on R (d)missing B, R on G

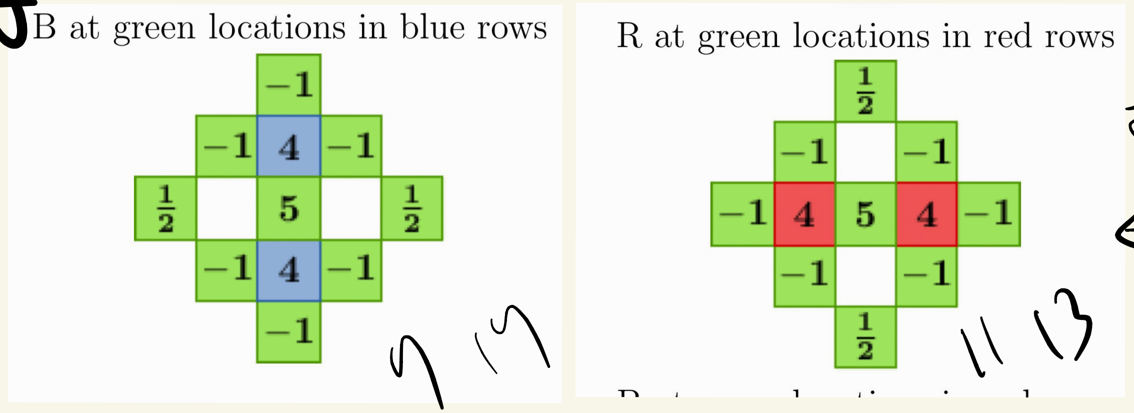
Take Fig. 5(a) for example, the center pixel is green. To obtain the missing color channels, the values of the blue channel and red channel of surrounding pixels are averaged and then rounded down. The calculation formulas are as below. The interpolation process for the case of Fig. 5(d) is similar to that of Fig. 5(a).

$$blue(129) = \left\lfloor \frac{blue(128) + blue(130)}{2} \right\rfloor$$

$$red(129) = \left\lfloor \frac{red(1) + red(257)}{2} \right\rfloor$$


reset
Initial
in_en
store_data
count=16383
case A
case B
row_count≠127
row_count=127
case C
row_count=127
case D
count=16383
Done

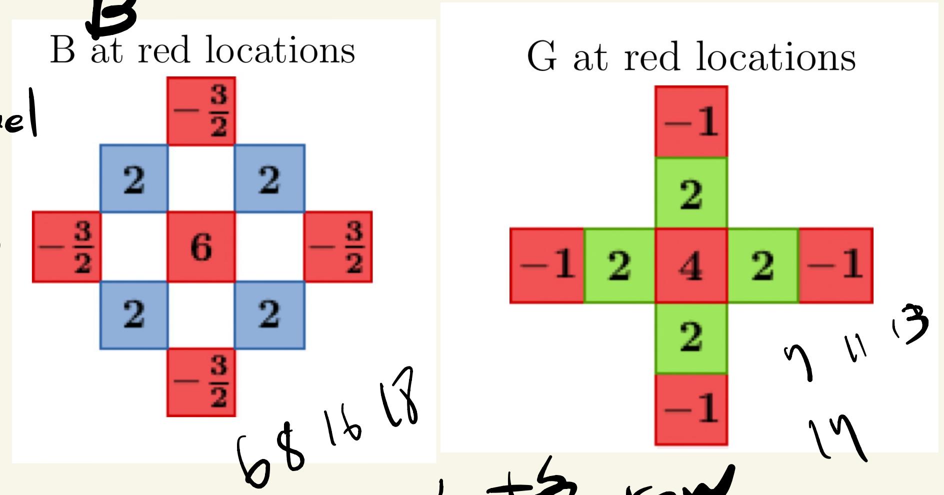
A


B at green locations in blue rows
R at green locations in red rows
7 17
11 13

换 pixel

$\leftrightarrow$

B

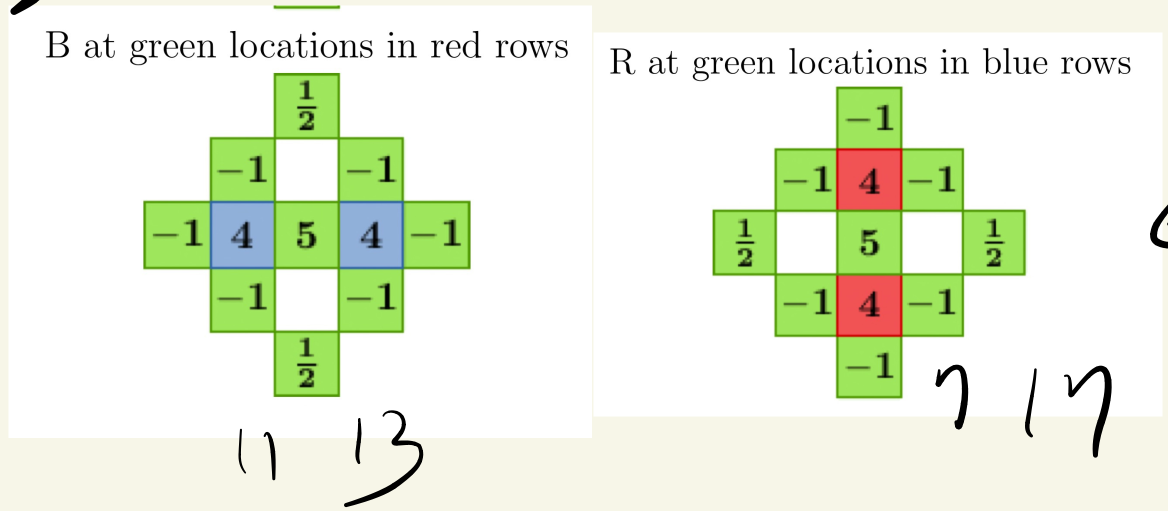


$\uparrow$ 换 row

$\downarrow$ 换 row

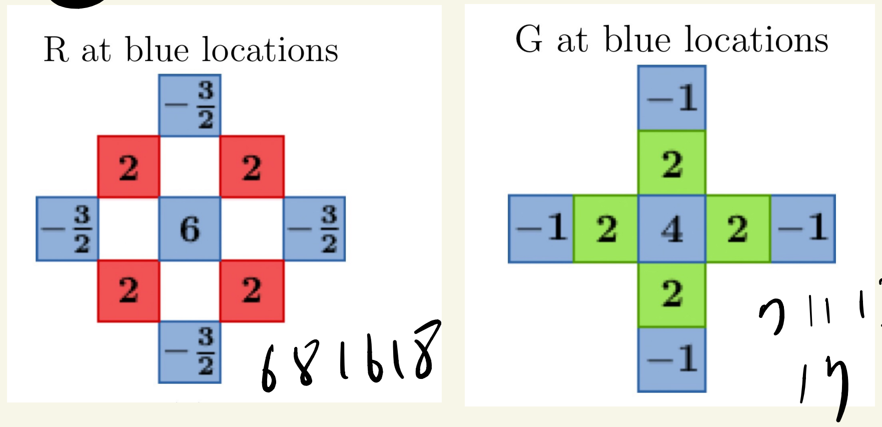
D

C


B at green locations in red rows
R at green locations in blue rows
11 13
7 17

换 pixel

$\leftrightarrow$



$$\frac{1}{8}\left(4k[7] + 5k[12] + \frac{1}{2}k[10] + \frac{1}{2}k[14] + 4k[17] - k[2] - k[6] - k[8]\right.$$

$$-k[16]-k[18]$$

$$=\frac{1}{16}\Big(8k[7]+10k[12]+k[10]+k[14]$$

$$+8k[17]-2k[2]-2k[6]-2k[8]$$

For the case of Fig. 5(b) and Fig. 5(c), there are four surrounding pixels having the required color channels of the center pixel. As a result, four values will be involved in the calculation formulas. That is, four values of green channel and four values of red/blue channel will be averaged and rounded down to interpolate the green channel and red/blue channel of the center pixel.

For convenience in implementation, edge pixels will not be included in the PSNR evaluation. Therefore, the interpolation of edge pixels can be skipped. After all pixels are interpolated, the full-color image should be written into the corresponding memory according to the color channel.

2.3.3 Interpolation Result Evaluation

After the simulation, the testfixture will generate raw image according to the values saved in the Red/Green/Blue memories. Then you can evaluate the interpolation results with the **psnr.exe** program provided by TA. This program will perform the following tasks:

(1) convert the raw images to PNG files and save them in "png" folder.
(2) calculate PSNR between the ground truth and the results.
(3) save the PSNR results in **psnr.txt** file.

3. Timing Specification

3.1 Bayer Pattern Input

After the system reset, the in_en signal will be pulled up. When the in_en signal is high, valid data of the input Bayer pattern will be sent consecutively with the data_in signal in each cycle. Since the data_in signal changes at the negative edge of the clk signal, you should retrieve data at the positive edge.

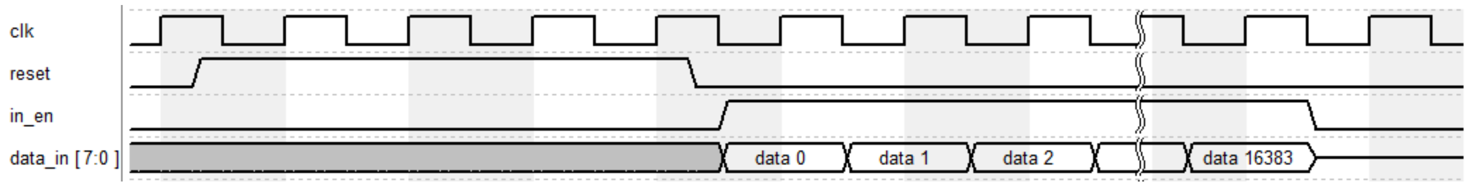


Figure 6. Timing diagram of input Bayer pattern sequence

3.2 Memory mapping and control

The arrangements of the Red/Green/Blue memories are the same as shown in Fig.4. All values in the three memories would be reset to zero at the start of the simulation. When the *wr* signal is set to high at the postive edge of clk, the value of the *wdata* port will be written to the corresponding address *addr* in memory at the negative edge.

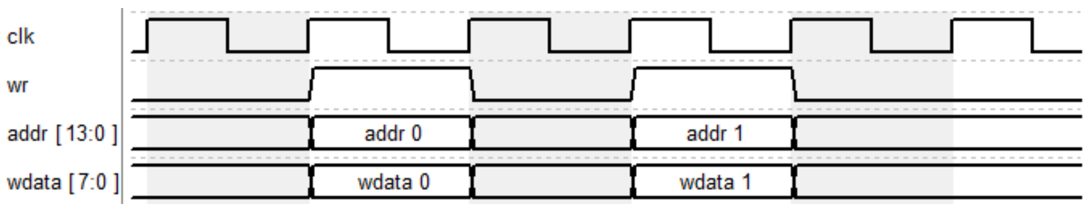


Figure 7. Timing diagram of writing memory

The Red/Green/Blue memories support read operation, so they can be used to store intermediate results. When the *wr* signal is set to low at the positive edge of *clk*, the testfixture will read data from memory according to the *addr* signal. The retrieved data can be obtained at the *rdata* port at the next positive edge of *clk*.

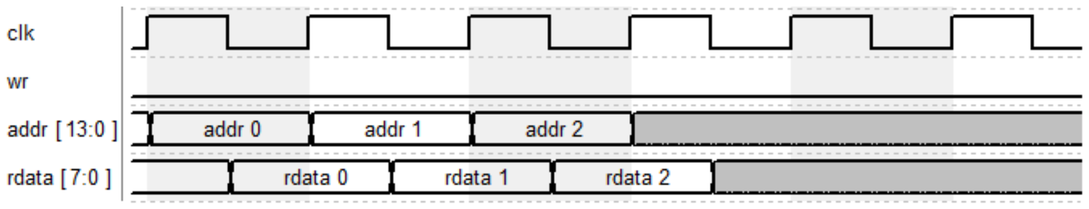


Figure 8. Timing diagram of reading memory

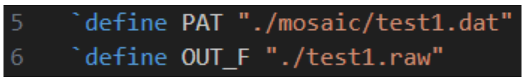
It's important to note that setting the *wr* signal to high means writing to memory. To avoid accidentally overwriting values of the memory, the *wr* signal should stay as low except doing the writing operation.

4. File Description

File Name	Description
demosaic.v	The top module of your design.
testfixture.v	The testbench file.
test1.dat ~ test6.dat	Input bayer pattern data.
test1.png ~ test6.png	Ground truth for evaluation.
psnr.exe	Executable file for calculating PSNR. If your OS is not Windows, you can contact TA for another executable file.

5. Scoring

For this homework, bilinear interpolation serves as the baseline. To get the score, the PSNR of your interpolation results should be no lower than baseline. Othewise, you will not get any functional simulation / gate-level simulation / performance score. The following table shows the baseline PSNR of each test image. Six images from the Kodak dataset are used for evaluation. The testfixture is designed to test one image at a time, and you can select the test image by modifying the path in lines 5~6. The number of test images should be 1~6.

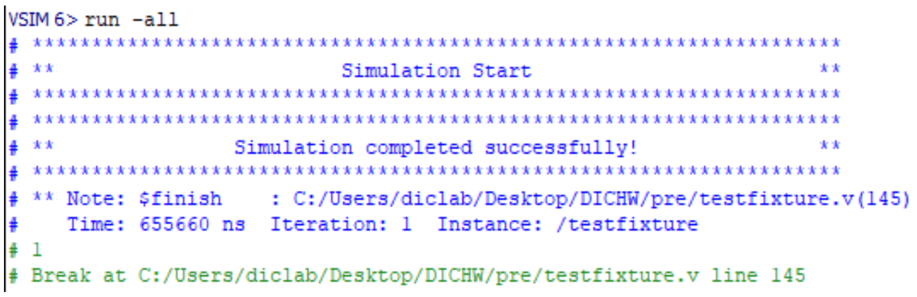


```
5   `define PAT "./mosaic/test1.dat"
6   `define OUT_F "./test1.raw"
```

Test image	Baseline PSNR
test1.png	25.29
test2.png	24.78
test3.png	29.13
test4.png	21.00
test5.png	21.98
test6.png	25.27

5.1 Functional simulation [40%]

All of the results should be generated correctly, and you will get the following message in ModelSim simulation. All your interpolation results must meet at least the baseline PSNR.



```
VSIM 6> run -all
# ********************************************************************
# **                        Simulation Start                        **
# ********************************************************************
# ********************************************************************
# **               Simulation completed successfully!               **
# ********************************************************************
# ** Note: $finish    : C:/Users/diclab/Desktop/DICHW/pre/testfixture.v(145)
#    Time: 655660 ns  Iteration: 1  Instance: /testfixture
# 1
# Break at C:/Users/diclab/Desktop/DICHW/pre/testfixture.v line 145
```

5.2 Gate-Level Simulation [40%]

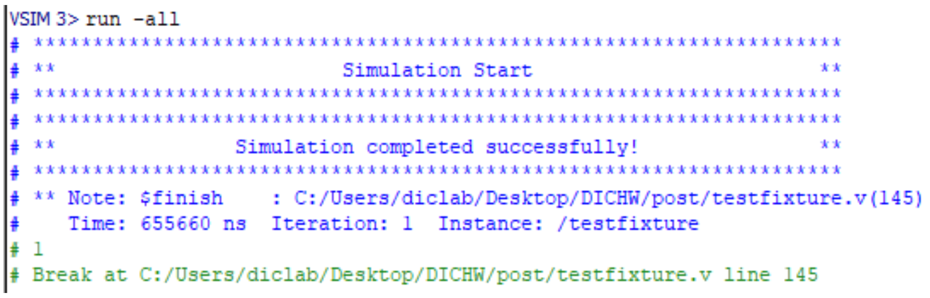
5.2.1 Synthesis

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

DEVICE：Cyclone IV E - EP4CE55F23A7

5.2.2 Simulation

All of the results should be generated correctly using demosaic.vo, and you will get the following message in ModelSim simulation. In this homework, the clock width is fixed at 40ns. The synthesized circuit has to be able to pass simulation with the specified clock width without timing violation. As with the rules of functional simulation, all your interpolation results must meet at least the baseline PSNR.



```
VSIM 3> run -all
# ********************************************************************
# **                        Simulation Start                        **
# ********************************************************************
# ********************************************************************
# **               Simulation completed successfully!               **
# ********************************************************************
# ** Note: $finish    : C:/Users/diclab/Desktop/DICHW/post/testfixture.v(145)
#    Time: 655660 ns  Iteration: 1  Instance: /testfixture
# 1
# Break at C:/Users/diclab/Desktop/DICHW/post/testfixture.v line 145
```

5.3 Performance [20%]

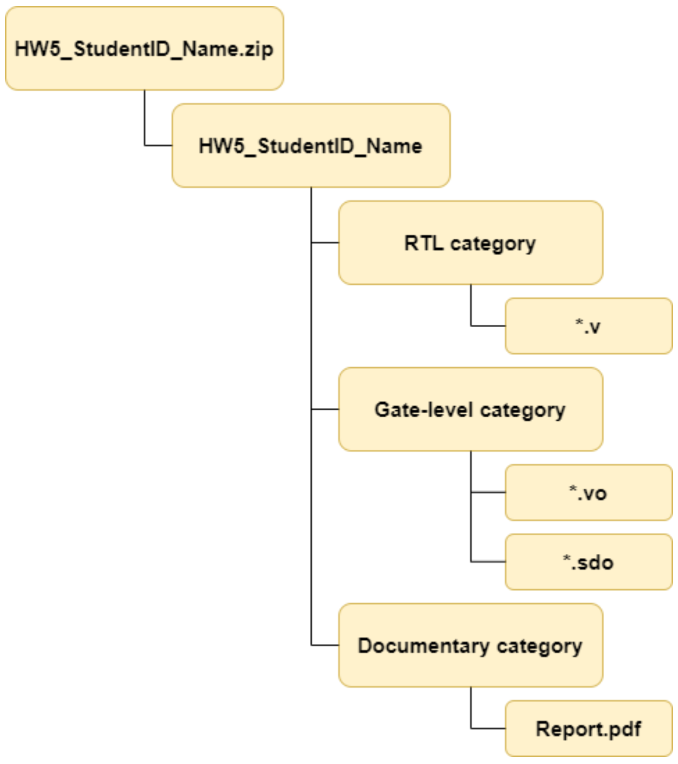
The performance is evaluated based on the average PSNR of the six test images. Your score will be determined by your ranking among all received homework submissions. Higher PSNR indicate better performance. Only designs that passed gate-level simulation and meet baseline PSNR will be considered in the ranking. Otherwise, you can’t get performance score.

6. Submission

6.1 Submitted files

You should classify your files into three directories and place them into a directory named HW5_studentID_name. Then compress the entire directory to .zip format. The naming rule is HW5_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).



6.2 Report file

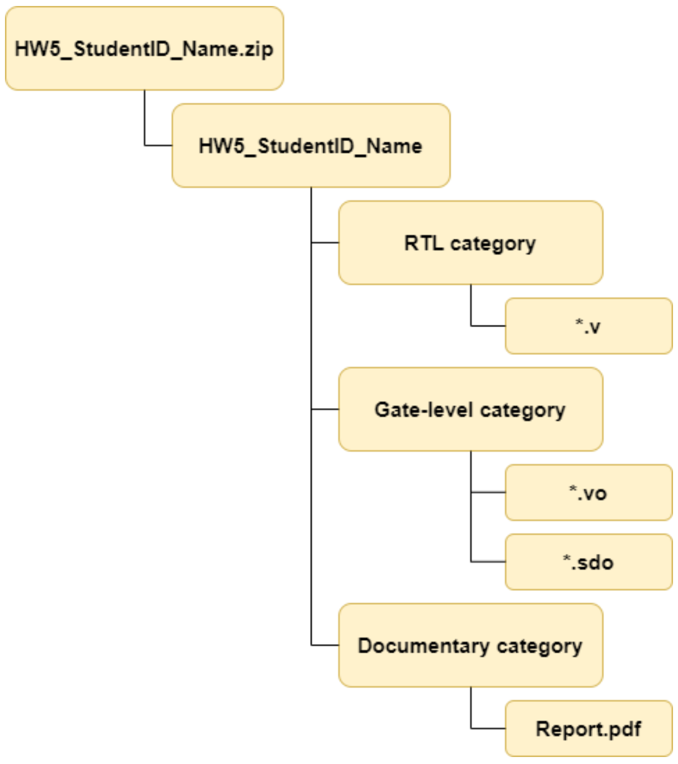
Please follow the spec of report. You are asked to describe how the circuit is designed as detailed as possible. If you use other demosaicing algorithm, you should describe it in detail too. Please fill in the fields of the PSNR of your interpolation results according to the results generated with the provided executable file. 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 HW5 in moodle.
Deadline: 2023/06/12 23:55

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

If you have any problem, please contact TA by email
p76114757@gs.ncku.edu.tw



6.2 Report file

Please follow the spec of report. You are asked to describe how the circuit is designed as detailed as possible. If you use other demosaicing algorithm, you should describe it in detail too. Please fill in the fields of the PSNR of your interpolation results according to the results generated with the provided executable file. 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 HW5 in moodle.
Deadline: 2023/06/12 23:55

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

If you have any problem, please contact TA by email
p76114757@gs.ncku.edu.tw