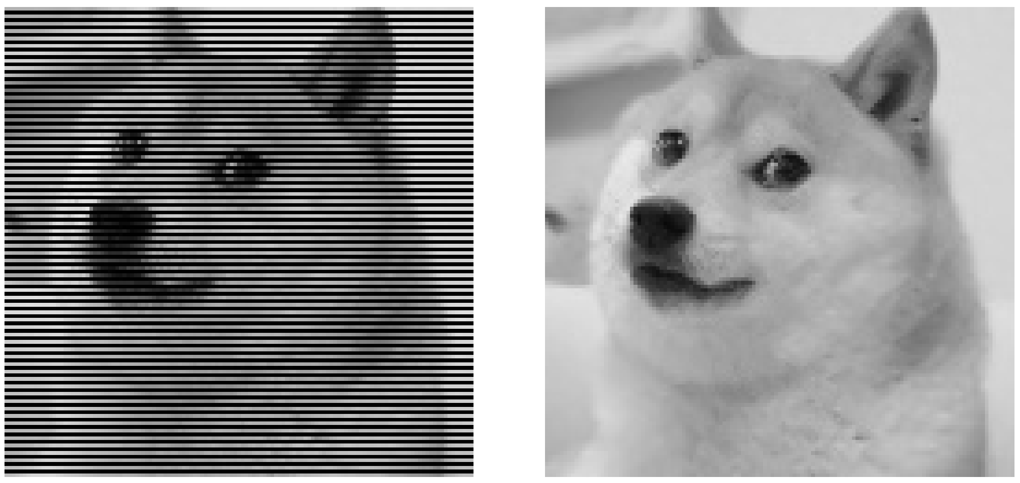
2022 Digital IC Design

Homework 4: Edge-Based Line Average interpolation

1 Introduction

The interlaced video comprises two types of fields in the sequence, one is the odd and another is the even field. The de-interlacing process is to convert the interlaced video into the non-interlaced form as shown in Fig. 1. The simplest method is intra-field interpolation, which use the existing pixels in the field to generate the empty lines. For instance, the empty lines can be filled via line doubling, which is quite easy to be implemented but the resulting image is not good enough in visual quality.

In this homework, you are asked to implement the Edge-Based Line Average interpolation algorithm. As the direction of edge is considered, the de-interlaced image has a better quality than merely doubling the existing lines.



(a) (b)

Fig. 1. The odd field (non-black row) of an interlaced video sequence (a) and the complete frame after de-interlacing (b).

2 Design Specifications

2.1 Block overview

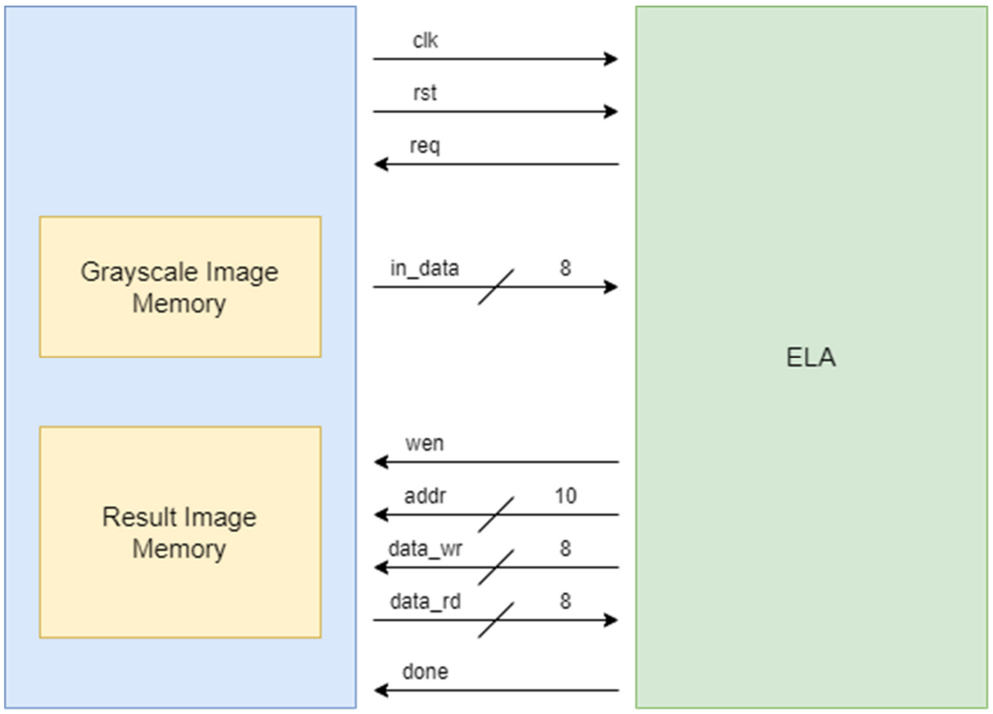


Fig. 2. System operation flow

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.
rst	I	1	Active-high asynchronous reset signal.
req	O	1	The request signal for a row of pixels.
in_data	I	8	The 8-bit input pixel data. When req signal is high, the testbench will send a row of image pixels from Grayscale Image Memory, one pixel per cycle.
wen	O	1	Result Image Memory write enable signal. When this signal is low (read), the data with address *addr* in Result Image 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 Image Memory.
addr	O	10	Result Image Memory read/write address. This signal indicates which address of the Result Image

			Memory will be read or written.
data_wr	O	8	Result Image Memory write data signal, which represents an 8 bits unsigned integer. The interpolation result of the ELA circuit is written to Result Image Memory through this signal.
data_rd	I	8	Result Image Memory read data signal, which represents an 8 bits unsigned integer. This signal is used to send the Result Image Memory data to ELA circuit.
done	O	1	After the interpolation of the whole image is completed and the result is completely output, setting this signal to high and the testbench will start checking if the result is correct.

2.3 Function Description

2.3.1 ELA Interpolation

The given test pattern in this homework is a 32x16 image, to which you need to design a circuit to perform edge-based line interpolation. Each row in the given test pattern is odd field, and you have to calculate the even row by edge-based line interpolation as shown in Fig. 3.

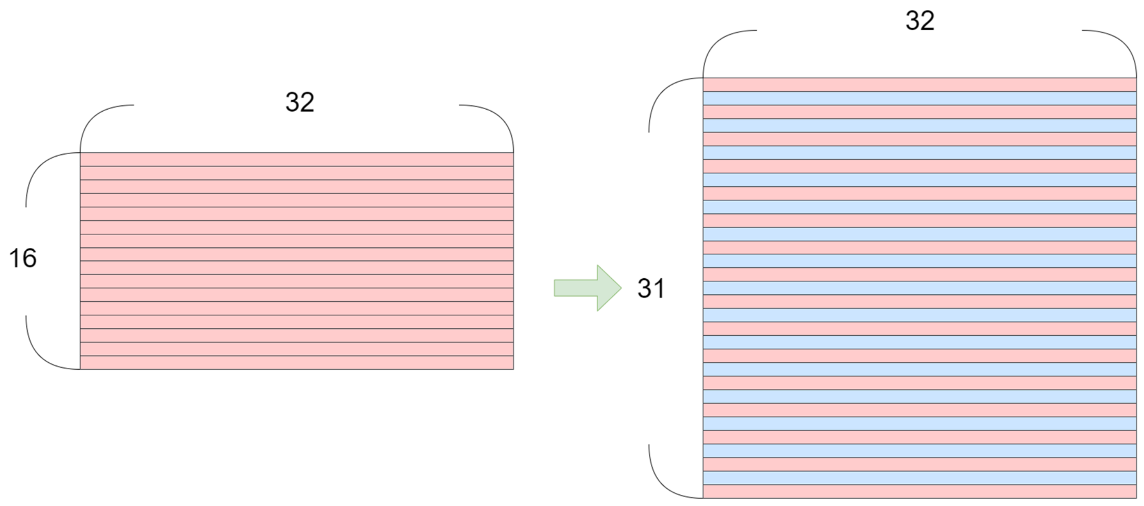


Fig. 3. The inter-frame interpolation. Red rows are odd field and blue rows are even field.

Assume that the pixel to be interpolated is located at coordinate (i , j) and pixels *a* to *f* are the neighboring points, which is shown in Fig. 4(a). First of all, three different directions at the interpolated position are calculated using

(1), and the value of interpolated pixel is obtained by (2). Note that, the fractional part can be ignored while calculating.

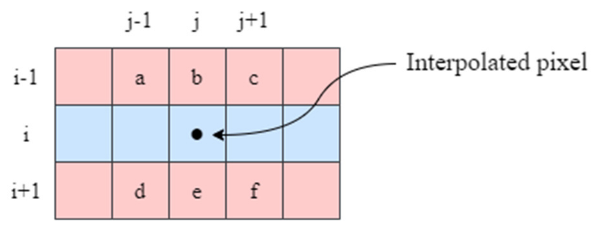


Fig. 4(a). The interpolated pixel and its neighboring points.

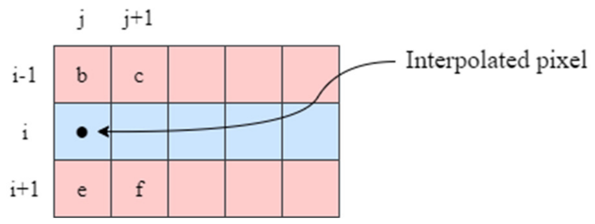


Fig. 4(b). The left boundary pixel and its neighboring points.

$$\begin{cases} D_1 = |a - f| \\ D_2 = |b - e| \\ D_3 = |c - d| \end{cases} \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ (1)$$

$$x(i,j) = \begin{cases} (a+f)/2\,, if \min(D_1, D_2, D_3) = D_1 \\ (b+e)/2\,, if \min(D_1, D_2, D_3) = D_2 \\ (c+d)/2\,, if \min(D_1, D_2, D_3) = D_3 \end{cases} \ldots\ldots\ldots\ldots\ldots\ldots(2)$$

If there are identical direction values, the priority of the three directions is $D_2 > D_1 > D_3$. For instance, $D_1=20$, $D_2=30$, $D_3=20$, then min $(D_1, D_2, D_3) = D_1$. The left and right boundary interpolation is fixed to (b+e)/2. Fig. 4(b) shows the example when the interpolation pixel is at the left boundary.

2.3.2 Data Input

The size of the input image in this system is 32x16, which is stored in the Grayscale Image Memory. The input order is from the first row to the last row and each row will only send one time. Setting the *req* signal to high to get one row of image data (as shown in Figure 5. t1 time point). Then, the pixel data will be sent into the ELA circuit by *in_data* signal at the negative edge of each cycle (as shown in figure 5. t2 time point). Please receive *in_data* at the positive edge of the clock cycle to avoid data unstable. After sending 32 pixels of a row, the testbench will stop sending data (as shown in figure 5. t3 time point) until the next high-level *req* signal (as shown in figure 5. t4 timing point).

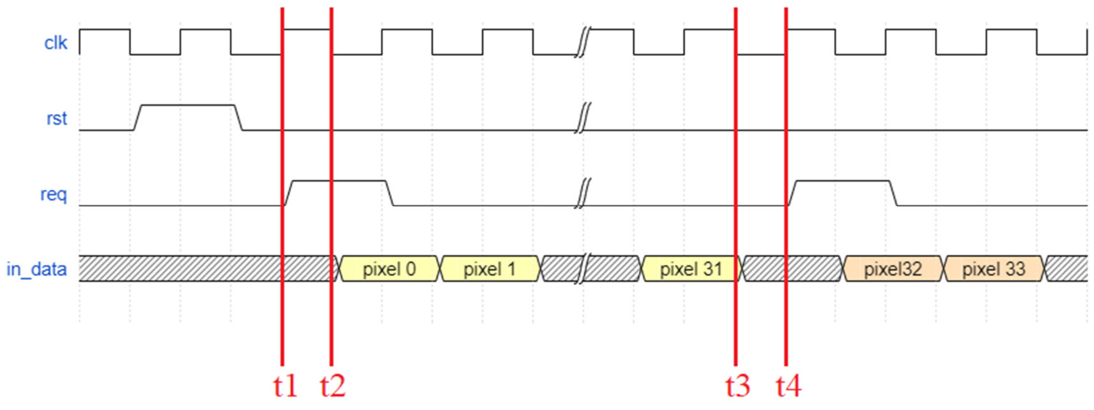


Fig. 5. Timing diagram while reading Grayscale Image Memory

2.3.3 Result Image Memory

The Result Image Memory is a 992x8 bit memory, which is used to store the result image of interpolation. The correspondence between each pixel of the image and memory arrangement is shown in Figure 6. Using *wen* signal to switch between reading mode and writing mode. Remember to set the *done* signal to high after writing all the result pixels to memory, then the testbench will check if the result is correct.

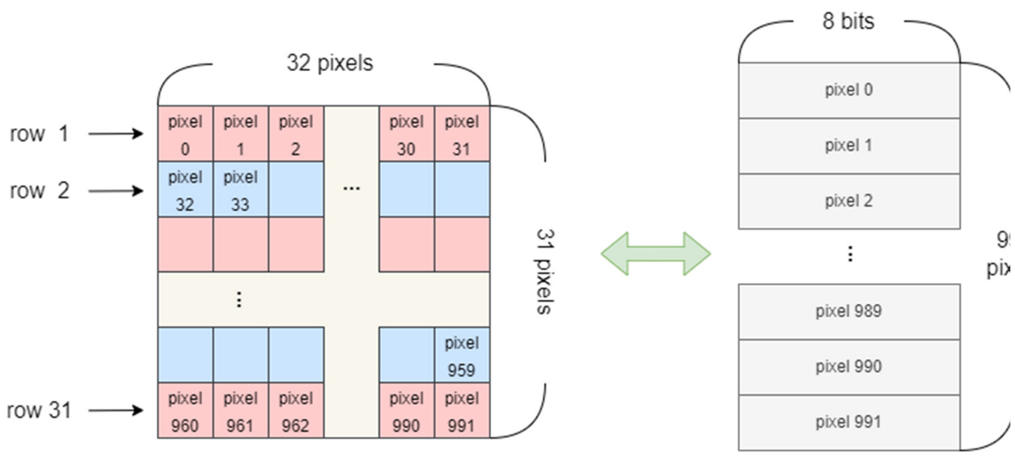


Fig. 6. Arrangement mapping of result image (left) and Result Image Memory (right).

In the reading mode, the *wen* signal should be set as low. The *addr* signal represents the memory address of result memory to read and the *data_rd* signal will provide the data. The timing diagram of result memory reading operation is shown in figure 7.

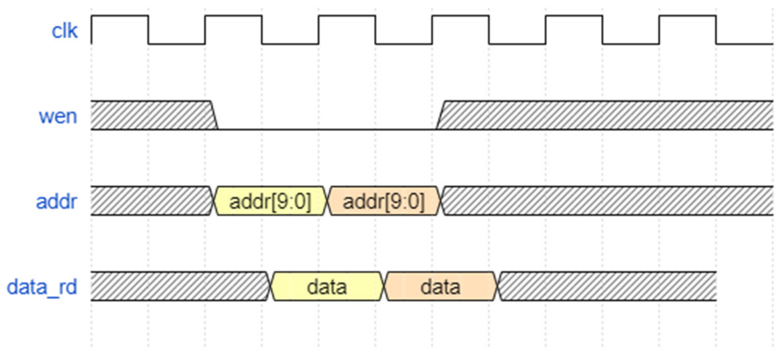


Fig. 7. Timing diagram of reading Result Image Memory.

In the writing mode, the *wen* signal should be set as high. The *addr* signal informs the memory address to be written and the *data_wr* signal should provide the writing data. The timing diagram of result memory writing operation is shown in figure 8.

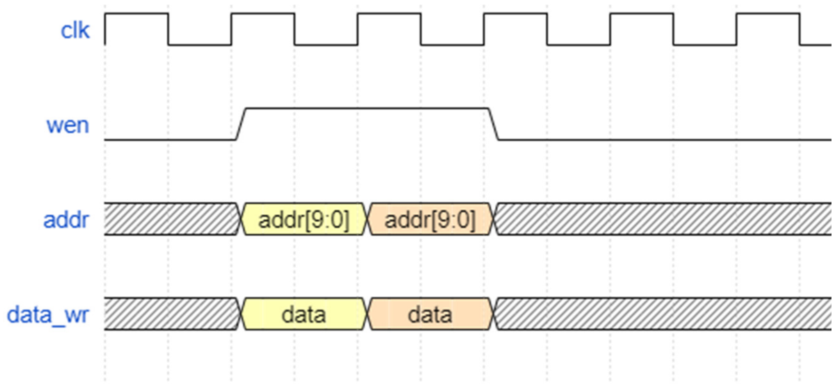


Fig. 8. Timing diagram of writing Result Image Memory.

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 should be ./image.jpg.
2. Convert RGB image to gray scale.
3. Resize the image into 32x31 (width is 32 and height is 31).
4. Remove the even row of pixels, then the image size will be 32x16.
5. Process edge-based line average interpolation to the image of which size is 32x16.
6. Output the image with size 32x16 and the interpolated 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 provided. The output files 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.
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.

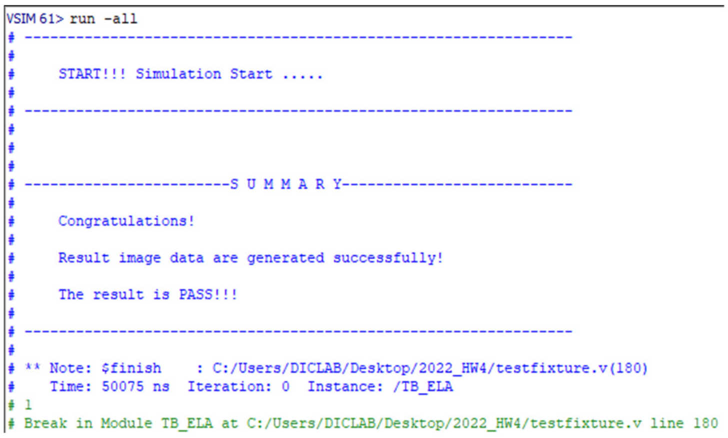
4 Scoring

4.1 Functional Simulation **[40%]**

The functional simulation will be partially scored as the following rules.

1. (10%) The odd row of the result image.
2. (10%) The boundary part of the even row of the result image.
3. (20%) The middle part of the even row of the result image.

You will not get the corresponding scores if you fail to pass the corresponding rule. You will get the message (shown in Fig. 9) in Modelsim simulation if passing all the test pattern.



```
VSIM 61> run -all
# ---------------------------------------------------------------
#
#     START!!! Simulation Start .....
#
# ---------------------------------------------------------------
#
#
#
# -----------------------S U M M A R Y--------------------------
#
#     Congratulations!
#
#     Result image data are generated successfully!
#
#     The result is PASS!!!
#
# ---------------------------------------------------------------
#
# ** Note: $finish    : C:/Users/DICLAB/Desktop/2022_HW4/testfixture.v(180)
#    Time: 50075 ns  Iteration: 0  Instance: /TB_ELA
# 1
# Break in Module TB_ELA at C:/Users/DICLAB/Desktop/2022_HW4/testfixture.v line 180
```

Fig. 9. Functional simulation pass message.

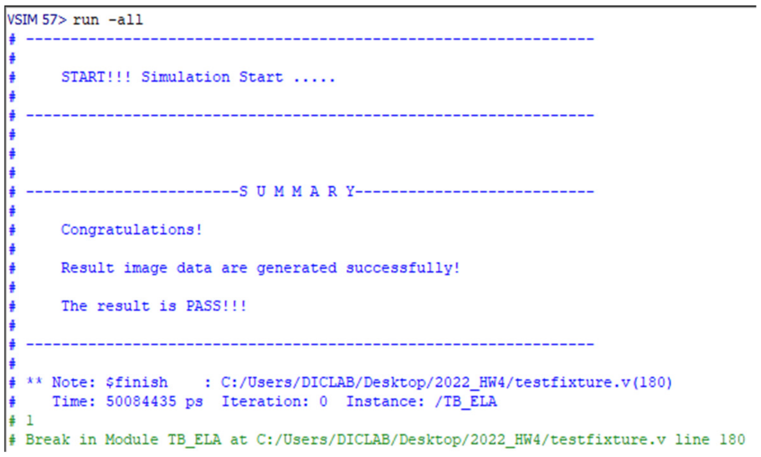
4.2 Gate Level Simulation **[20%]**

4.2.1 Synthesis

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

4.2.2 Simulation

All of the result should be generated correctly using *ELA.vo*, and you will get the following message (shown in Fig 10.) in ModelSim simulation. You will not get any score of gate level simulation if you fail to pass the test pattern.



```
VSIM 57> run -all
# ----------------------------------------------------------------
#
#     START!!! Simulation Start .....
#
# ----------------------------------------------------------------
#
#
#
# -----------------------S U M M A R Y--------------------------
#
#     Congratulations!
#
#     Result image data are generated successfully!
#
#     The result is PASS!!!
#
# ----------------------------------------------------------------
#
# ** Note: $finish    : C:/Users/DICLAB/Desktop/2022_HW4/testfixture.v(180)
#    Time: 50084435 ps  Iteration: 0  Instance: /TB_ELA
# 1
# Break in Module TB_ELA at C:/Users/DICLAB/Desktop/2022_HW4/testfixture.v line 180
```

Fig. 10. Gate level simulation pass message.

Device：Cyclone II EP2C70F896C8

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 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 softwate verification if you fail to pass the hidden test pattern.

5 Submission

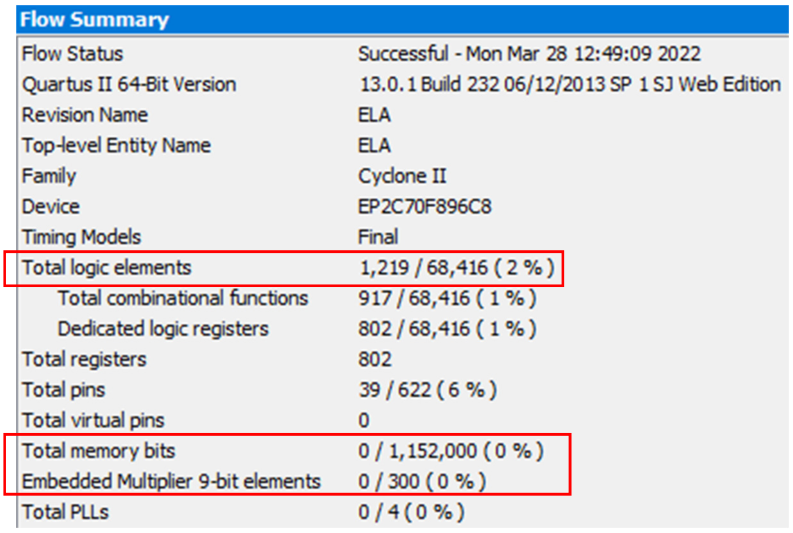
5.1 Submitted files

You should classify your files into four directories as the table below 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 or file hierarchy is wrong, you will lose 5 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.
Readme.txt	(optional)

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 (as show in Fig. 11.) 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 - Mon Mar 28 12:49:09 2022
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
Revision Name	ELA
Top-level Entity Name	ELA
Family	Cyclone II
Device	EP2C70F896C8
Timing Models	Final
Total logic elements	1,219 / 68,416 (2 %)
Total combinational functions	917 / 68,416 (1 %)
Dedicated logic registers	802 / 68,416 (1 %)
Total registers	802
Total pins	39 / 622 (6 %)
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 %)

Fig. 11. Example of flow summary of synthesized design.

5.3 Note

In this homework, you are allowed to modify the defined *CYCLE* and *End_CYCLE* in testbench file. *CYCLE* decides the clock width in your design, and *End_CYCLE* decides the maximum cycles your circuit take to complete simulation. Please do not modify any other content of testbench.

Please submit your .zip file to folder HW4 in moodle.
Deadline: 2022/5/27 16:59

If you have any problem, please contact TA by email.
way271283@gmail.com