2018 Digital IC Design

Homework 3: Adaptive Threshold Engine

1. Introduction

Threshold is the simplest method of image segmentation. From a grayscale image, threshold can be used to create binary images

Please design an Adaptive Threshold Engine (ATE). As shown in Fig 1, the function of the ATE circuit is used to separate a grayscale image from the foreground image. The system block diagram is shown in Fig 2. Its detailed specifications will be described later. Each input and output signal refers to **Table 1**.

Fig 1. The function of Adaptive Threshold Engine

2. Design Specifications

2.1 Block Overview

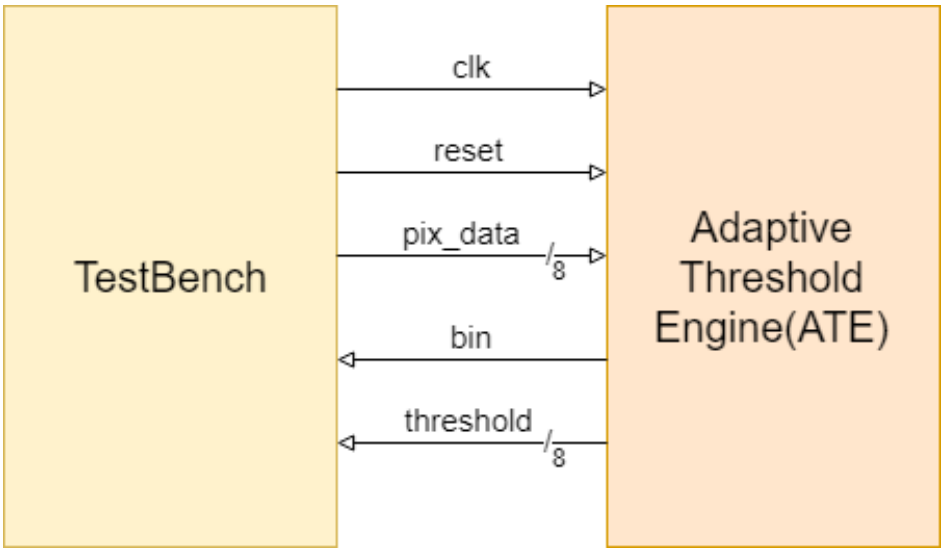


Fig 2. Block Overview

2.2 I/O Interface

Table 1. I/O Interface

Signal Name	I/O	Width	Description
clk	I	1	clock for the computational system
reset	I	1	reset the state of the computational system when it asserts
pix_data	I	8	the 8-bit input pixel data
bin	O	1	each pixel data is converted to 1-bit data by threshold
threshold	O	8	threshold data for single region

2.3 System Description

2.3.1 Image input

The size of input image is 48X32 (6X4 block). Each block has 8X8 points. The image input order is sequentially, as shown in the block number sequence of **Fig 3.**

When performing Threshold processing, it takes the block-base as a unit. The leftmost and rightmost lines do not need to be processed, so their bin and threshold are all outputs 0. As shown in Fig 3, blocks 0, 6, 12, 18 and The 5, 11, 17, 23 blocks do not need to be processed.

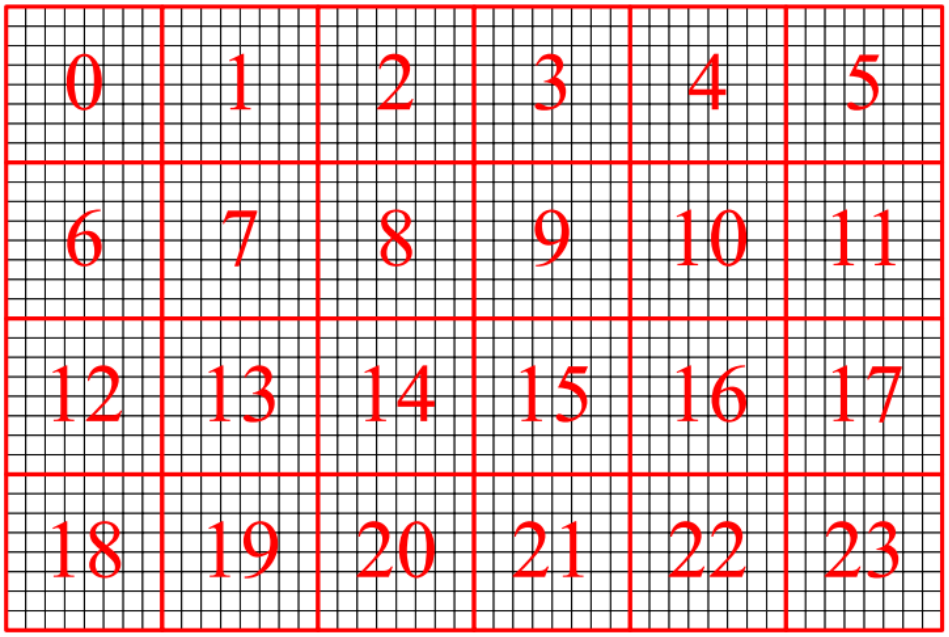


Fig 3. Input image block

2.3.2 Adaptive Threshold

The purpose of threshold is to distinguish the foreground and background of a grayscale image. For each point of the image block, if it is greater than or equal to the threshold value of the block, it outputs 1 and vice versa. The Adaptive threshold indicates that this threshold is calculated.

The threshold calculation method for this ATE circuit uses the average of the maximum and minimum values in a single 8x8 block, i.e. threshold=(Max+Min)/2, and if the threshold is a floating point number, **the unconditional carry is taken**.

For each point:

bin=1 if pix_data >= threshold

bin=0 if pix_data < threshold

In **Fig 4**, for example, the maximum value of the block is 192, and the minimum value is 48. Then the threshold = (192+48)/2=120, so the output is the right picture.

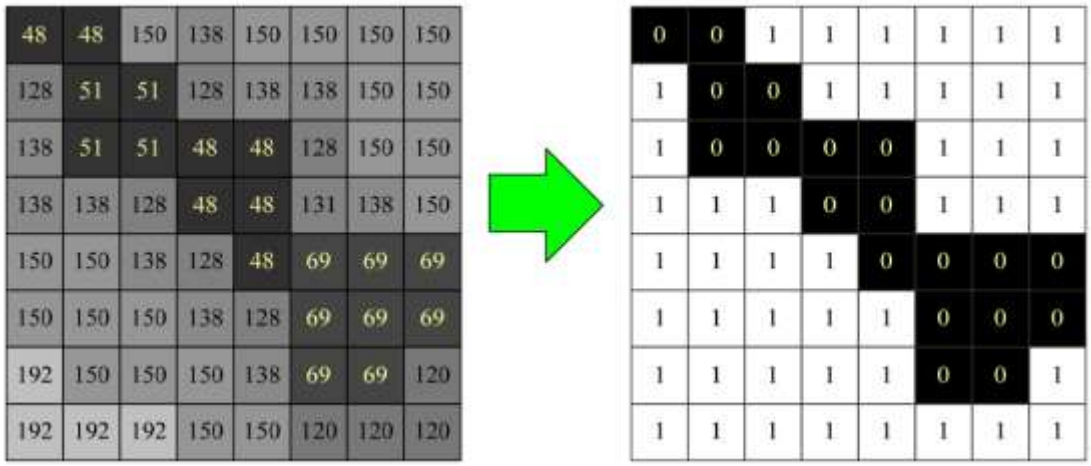


Fig 4. Single block threshold

2.4 Timing Diagram

2.4.1 Output Order

The test circuit will be reset at the beginning. After the reset is completed, the entire image content will be input in an uninterrupted manner. The order of the input blocks is from left to right, from top to bottom. In a single block, the value of each point

is also from left to right and from top to bottom. The order of output is the same as the order of input, but the output will add an output delay time one block later than the input, and the output will be uninterrupted.

As shown in **Fig 5**, the output delay is set to 1 cycle. When the test circuit inputs the content of the first block, the ATE sends the calculation result of the first block at the next cycle.

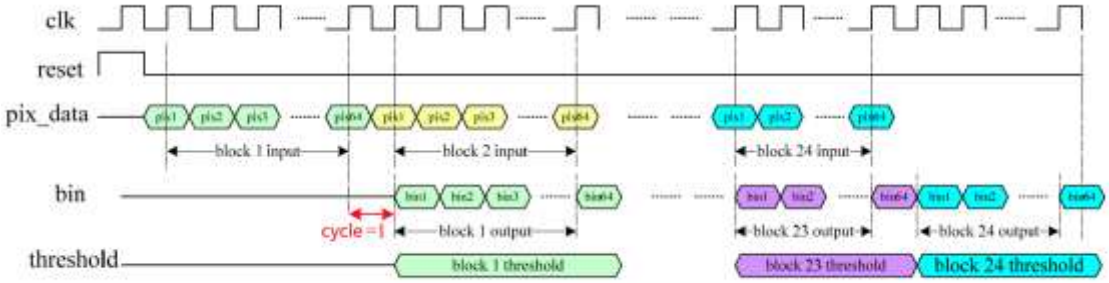


Fig 5. Output order

2.4.2 Output timing

The input and output timing specifications of this circuit are as follows:

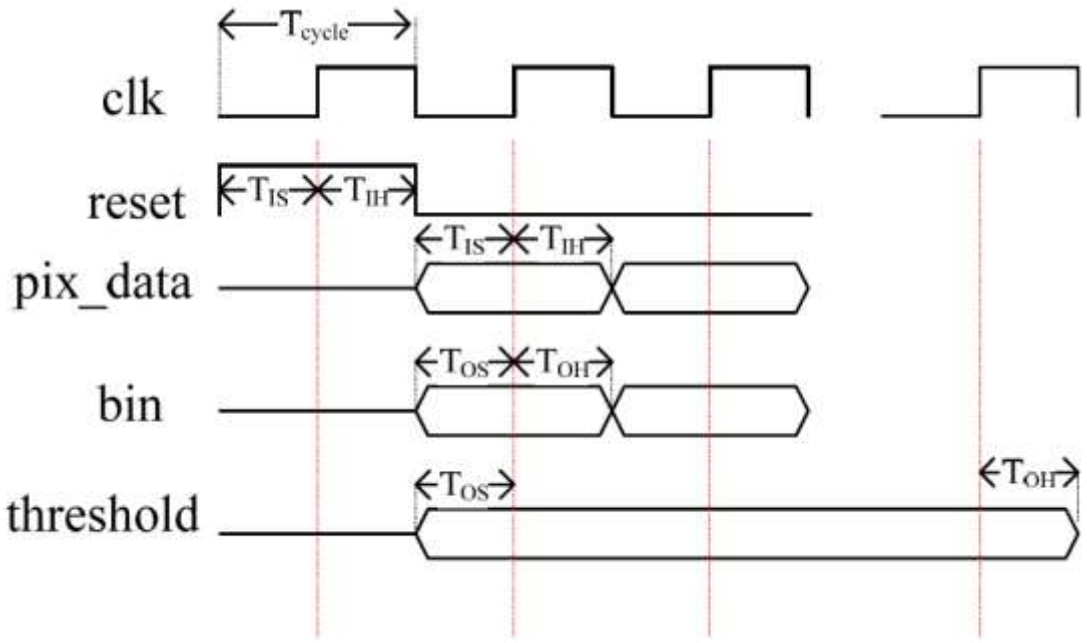


Fig 6. Timing definition

Table 2

Symbol	Description	Value
T_{cycle}	Clock cycle	User definition

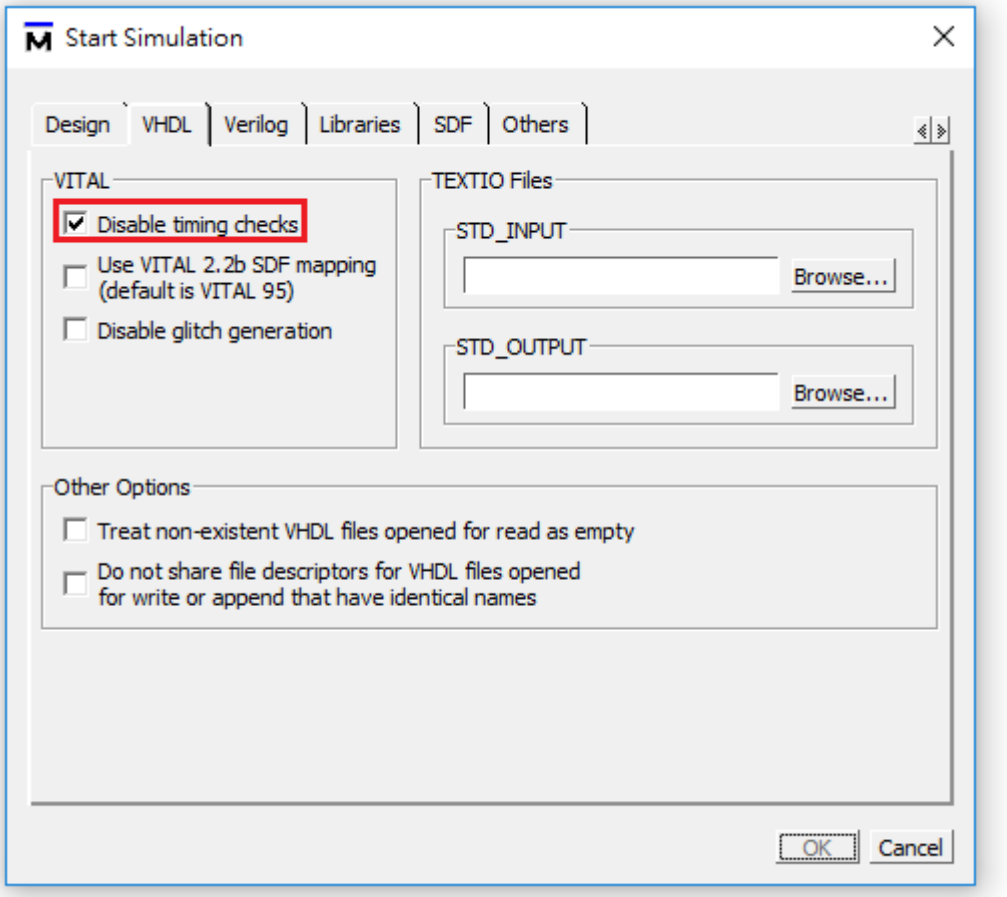
T_{IS}	Setup time	1/2 clock cycle
T_{IH}	Hold time	1/2 clock cycle
T_{OS}	Setup time	1/2 clock cycle
T_{OH}	Hold time	1/2 clock cycle

3. Scoring

3.1 Functional Simulation (pre-sim) **[60%]**

All of the result should be generated correctly, and you will get the following message in ModelSim simulation. You can turn off the timing check in pre-sim only.

```
#                               ,-.__.-,
#                              (``-''-//).___..--'''`-._
#                               `6_ 6  )   `-.  (     ).`-.__.`)
#                               (_Y_.)'  ._   )  `._ `. ``-..-'
#                             _..`--'_..-_/  /--'_.' ,'
#                            (il),-''  (li),'  ((!.-'
#   Congratulations !
#   Simulation Complete!!
#
# ** Note: $finish    : D:/Ian/IC/class/New_HW3/icc2010ucb/presim/testfixture.v(168)
#    Time: 16025 ns  Iteration: 0  Instance: /testfixture
```



3.2 Gate-Level Simulation (post-sim) **[20%]**

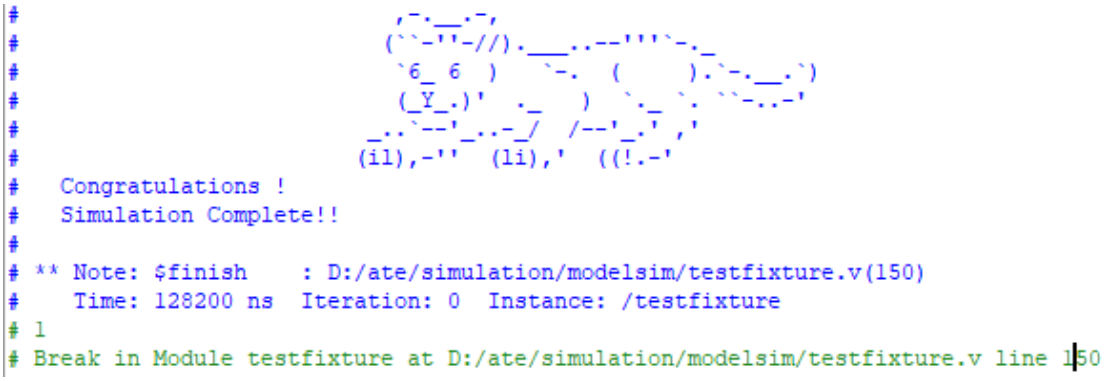
3.2.1 Synthesis

Your code should be synthesizable. After synthesizing in Quartus, the file named ate.vo and ate.sdo will be obtained.

Device : Cyclone II EP2C70F896C8

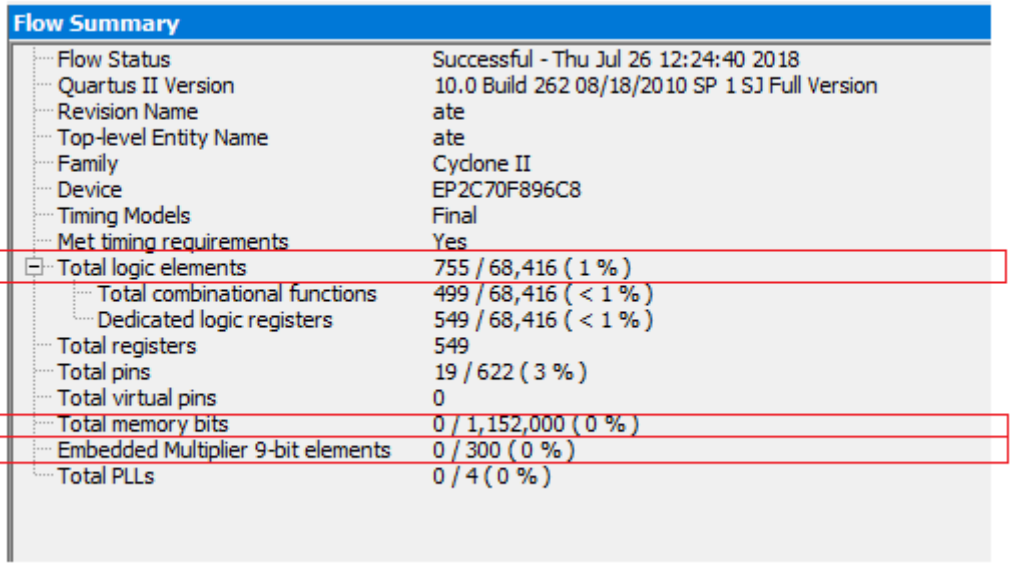
3.2.2 Simulation

All of the result should be generated correctly using ate.vo and ate.sdo, and you will get the following message in ModelSim simulation. (There should be no setup or hold time violations.)

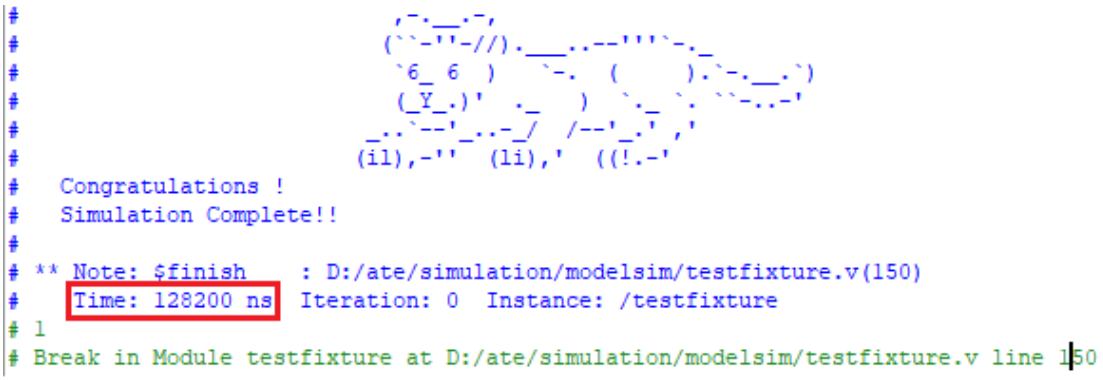


3.3 Performance **[20%]**

The performance is scored by the logic elements you used and the simulation time in post-sim. The scoring equation is *(Total logic elements + total memory bit + 9*embedded multiplier 9-bit element)* × *(total simulation time in ns)*. (The smaller the better).



Flow Summary	
Flow Status	Successful - Thu Jul 26 12:24:40 2018
Quartus II Version	10.0 Build 262 08/18/2010 SP 1 SJ Full Version
Revision Name	ate
Top-level Entity Name	ate
Family	Cyclone II
Device	EP2C70F896C8
Timing Models	Final
Met timing requirements	Yes
Total logic elements	755 / 68,416 (1 %)
Total combinational functions	499 / 68,416 (< 1 %)
Dedicated logic registers	549 / 68,416 (< 1 %)
Total registers	549
Total pins	19 / 622 (3 %)
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 %)



```
#                                ,-.__.-,
#                               (``-''-//).___..--'''`-._
#                                `6_ 6  )   `-.  (     ).`-.__.`)
#                                (_Y_.)'  ._   )  `._ `. ``-..-'
#                              _..`--'_..-_/  /--'_.' ,'
#                             (il),-''  (li),'  ((!.-'
#   Congratulations !
#   Simulation Complete!!
#
# ** Note: $finish    : D:/ate/simulation/modelsim/testfixture.v(150)
#    Time: 128200 ns  Iteration: 0  Instance: /testfixture
# 1
# Break in Module testfixture at D:/ate/simulation/modelsim/testfixture.v line 150
```

4. Submission

4.1 Submitted files

You should classify your files into three directories and compressed to .zip format. The naming rule is **HW3_*studentID*_ *version*.zip**. The *vision* is v1 for the first submission, and v2, v3… for the revisions.

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).

4.2 Report file

You have to describe how the circuit is designed as detailed as possible, and the flow summary result and simulation results are necessary. **Please follow the specification in appendix.**

4.3 Please submit your .zip file to folder HW3 in the ftp site.

Deadline: 107/11/23 23:55

ftp : 140.116.245.92

Username : ic_design

Password : ic_design

5. If you have any problem, please contact the TA by email:

陳威廷: weiting84610@gmail.com

方宏育: ian840512@gmail.com