FDE Lab Report

Sun Zhongji 17300750004

School of Microelectronics

Overview

The report is aimed at presenting what I have learned and practiced from the course and experiments with the FDE development board. Some problems were encountered while more lessons were learned through the process of handling them. Therefore, they will be shown in this report, which consists of 5 parts, including Development Environment, Lab 0, Lab 4 for inverter rewritten and image process and Problems & Improvements.

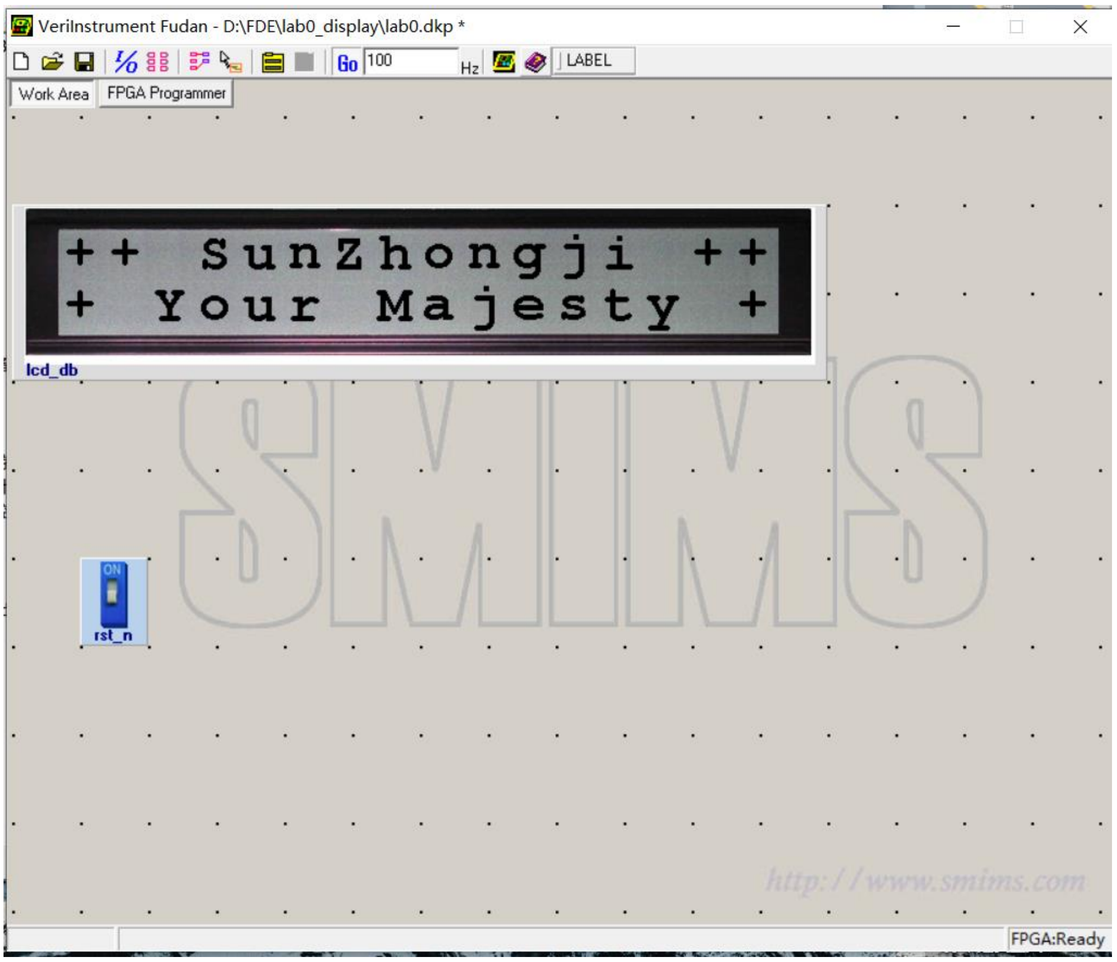
Development Environment

It is quite important to notify the development environment because most problems originate from the change of version of Windows and Visual Studio.

- Windows 10 of 64 bit
- Visual Studio 2019 for 64 bit (32-bit version used infrequently and will be <u>underlined</u>)

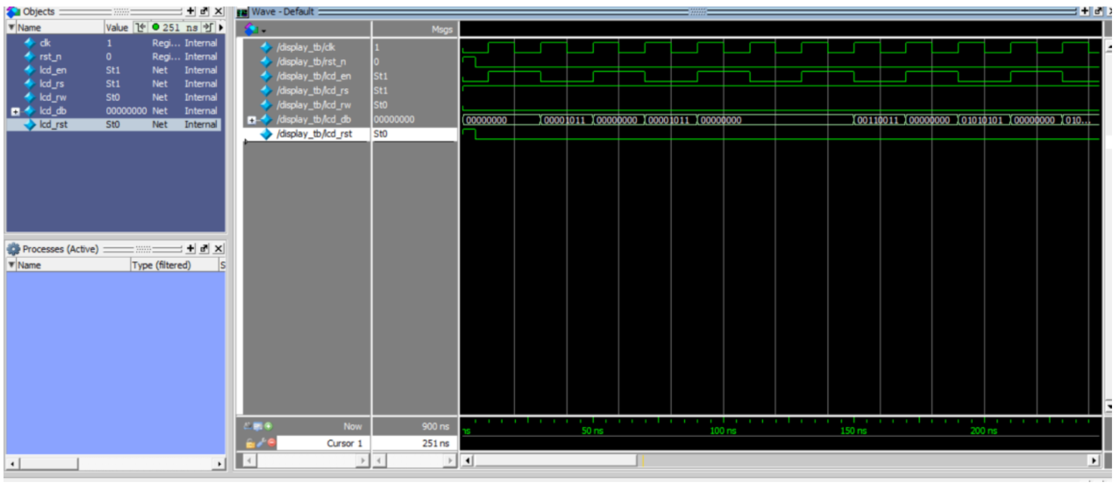
Lab 0

Lab 0 is a simple one which orient us into the learning of FDE. The outcome is originally a caption of "Welcome to Fudan University [Your name]". But to show what I have done, I changed it to "[Your name] Your Majesty Welcome to FPGA WORLD".



```
++  SunZhongji  ++
+  Your Majesty  +
```

Running Lab 0



Testbench for Lab 0

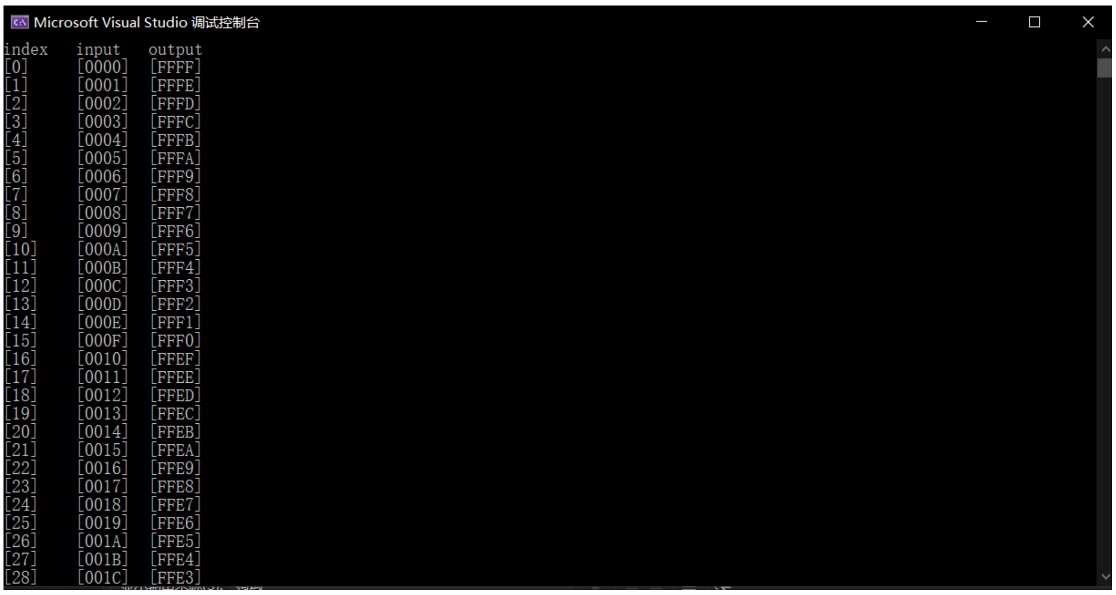
Lab 4 for inverter rewritten

This part is a quite tricky one because most problems are shown here.

First, I strictly complied to what was told in the documentation, but turned out useless and many errors were exposed. Then I came up with what was done in my graduation design where I had rewritten the Dynamic Link Library of FDE.

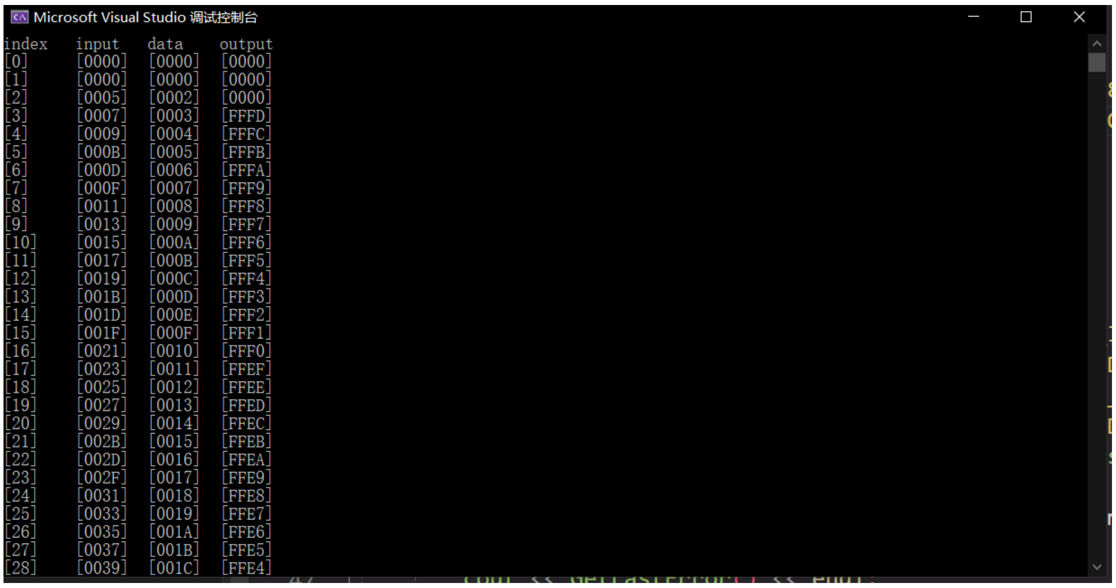
Therefore, I turned to it and rewrote the example from top to bottom. It turned out to be successful with few shortcomings that could be ignored. However, to make the report complete and easy to follow, I am going to present them in detail.

- The rewritten DLL uses different interface functions with the beginning of “IO” in their names while the old ones uses “APP”.
- The rewritten DLL supports different platforms and OS, including Windows, MacOS and Linux, with both 64- and 32-bit version.
- The rewritten DLL does not use the 1024×64 FIFO that the old one uses, so the Verilog code can be simpler but a bit worse in performance.
- The rewritten DLL enables the output a clock cycle after the input, so it needs some more work in software programming tackling the delay of the output.
- The rewritten DLL relies on an updated USB protocol so it is a bit annoying to switch between the old USB driver and the new one.
- The rewritten DLL is not highly dependent on the given constraint xml file which is fixed and not fully illustrated so that we can use HDLAutoAssign as if the Verilog code is as normal as that in other hardware-only projects.



```
Microsoft Visual Studio 调试控制台
index   input   output
[0]     [0000]  [FFFF]
[1]     [0001]  [FFFE]
[2]     [0002]  [FFFD]
[3]     [0003]  [FFFC]
[4]     [0004]  [FFFB]
[5]     [0005]  [FFFA]
[6]     [0006]  [FFF9]
[7]     [0007]  [FFF8]
[8]     [0008]  [FFF7]
[9]     [0009]  [FFF6]
[10]    [000A]  [FFF5]
[11]    [000B]  [FFF4]
[12]    [000C]  [FFF3]
[13]    [000D]  [FFF2]
[14]    [000E]  [FFF1]
[15]    [000F]  [FFF0]
[16]    [0010]  [FFEF]
[17]    [0011]  [FFEE]
[18]    [0012]  [FFED]
[19]    [0013]  [FFEC]
[20]    [0014]  [FFEB]
[21]    [0015]  [FFEA]
[22]    [0016]  [FFE9]
[23]    [0017]  [FFE8]
[24]    [0018]  [FFE7]
[25]    [0019]  [FFE6]
[26]    [001A]  [FFE5]
[27]    [001B]  [FFE4]
[28]    [001C]  [FFE3]
```

Running Documentation of Lab 4



```
Microsoft Visual Studio 调试控制台
index   input   data    output
[0]     [0000]  [0000]  [0000]
[1]     [0000]  [0000]  [0000]
[2]     [0005]  [0002]  [0000]
[3]     [0007]  [0003]  [FFFD]
[4]     [0009]  [0004]  [FFFC]
[5]     [000B]  [0005]  [FFFB]
[6]     [000D]  [0006]  [FFFA]
[7]     [000F]  [0007]  [FFF9]
[8]     [0011]  [0008]  [FFF8]
[9]     [0013]  [0009]  [FFF7]
[10]    [0015]  [000A]  [FFF6]
[11]    [0017]  [000B]  [FFF5]
[12]    [0019]  [000C]  [FFF4]
[13]    [001B]  [000D]  [FFF3]
[14]    [001D]  [000E]  [FFF2]
[15]    [001F]  [000F]  [FFF1]
[16]    [0021]  [0010]  [FFF0]
[17]    [0023]  [0011]  [FFEF]
[18]    [0025]  [0012]  [FFEE]
[19]    [0027]  [0013]  [FFED]
[20]    [0029]  [0014]  [FFEC]
[21]    [002B]  [0015]  [FFEB]
[22]    [002D]  [0016]  [FFEA]
[23]    [002F]  [0017]  [FFE9]
[24]    [0031]  [0018]  [FFE8]
[25]    [0033]  [0019]  [FFE7]
[26]    [0035]  [001A]  [FFE6]
[27]    [0037]  [001B]  [FFE5]
[28]    [0039]  [001C]  [FFE4]
```

Running Self-made Lab 4

Lab4 for image process

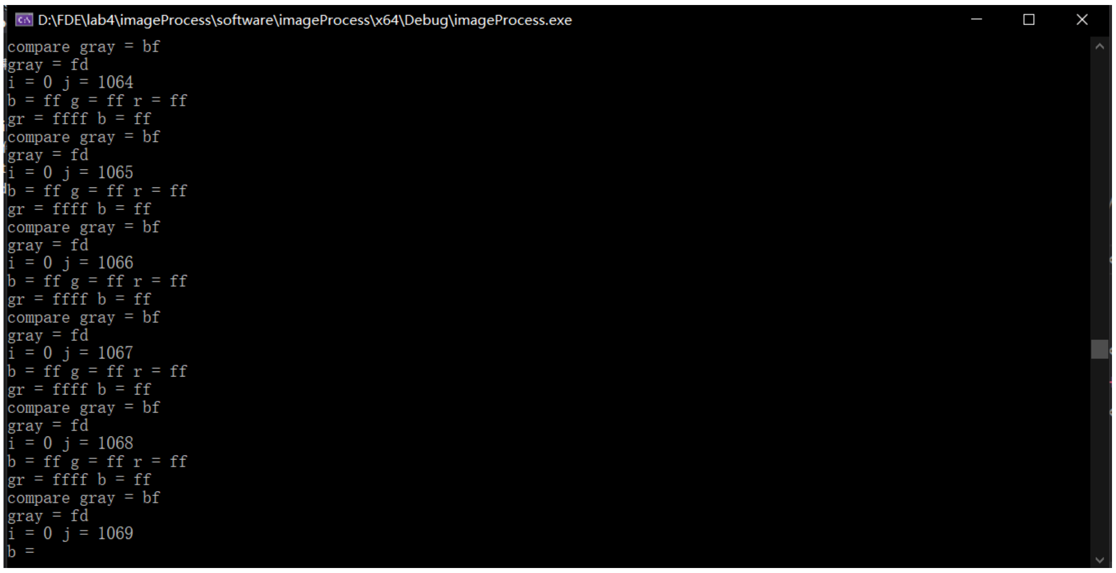
After successful implementation of rewriting inverter example, I proceeded to image process using FDE board. It was implemented by someone in 前人 PJ but I did a bit more than that.

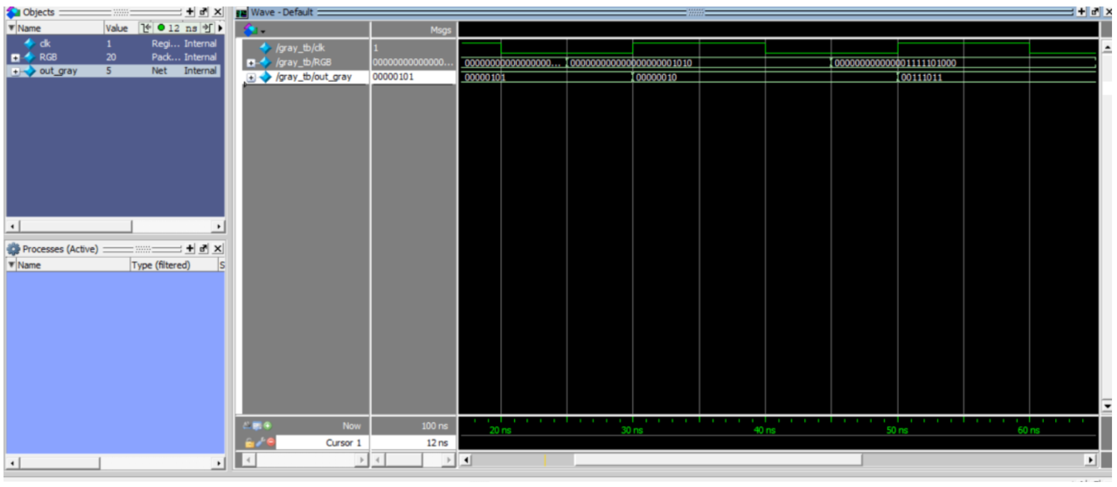
The previous project created a mean filter and did binarization of image using old DLL. In comparison, I used the new DLL and deprecated the mean filter for its not-so-good outcome. Moreover, I relied less on OpenCV. The previous project used OpenCV to read in the image and created the grayscale image of it using software while I created the grayscale image using the hardware of FDE board.

Diving into the details of grayscale transformation, I put a weight on RGB. Red is 0.25. Green is 0.5. Blue is 0.25. The weight set is said to be close to human's eye attention on three color channels. It takes some time to combine three 8-bit number into a 24-bit number and transmit into FDE board.

Binarization is easier since a threshold is set to grayscale that if it is higher than the threshold it becomes white, otherwise black.

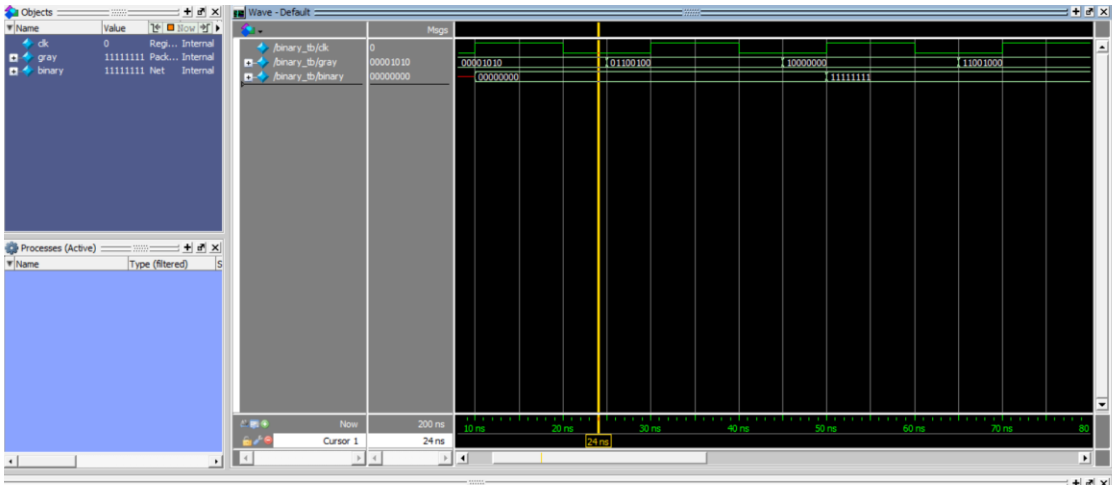
I used some of my pictures to test the effects of them. To show the normality, I chose one real-life landscape and one Japanese-anime. It seems that the anime picture works well because its color and edge are easy to tell while real-life pictures should deserve more work on processing color styles and edges.


```
D:\FDE\lab4\imageProcess\software\imageProcess\x64\Debug\imageProcess.exe
compare gray = bf
gray = fd
i = 0 j = 1064
b = ff g = ff r = ff
gr = ffff b = ff
compare gray = bf
gray = fd
i = 0 j = 1065
b = ff g = ff r = ff
gr = ffff b = ff
compare gray = bf
gray = fd
i = 0 j = 1066
b = ff g = ff r = ff
gr = ffff b = ff
compare gray = bf
gray = fd
i = 0 j = 1067
b = ff g = ff r = ff
gr = ffff b = ff
compare gray = bf
gray = fd
i = 0 j = 1068
b = ff g = ff r = ff
gr = ffff b = ff
compare gray = bf
gray = fd
i = 0 j = 1069
b =
```
Running Image Process



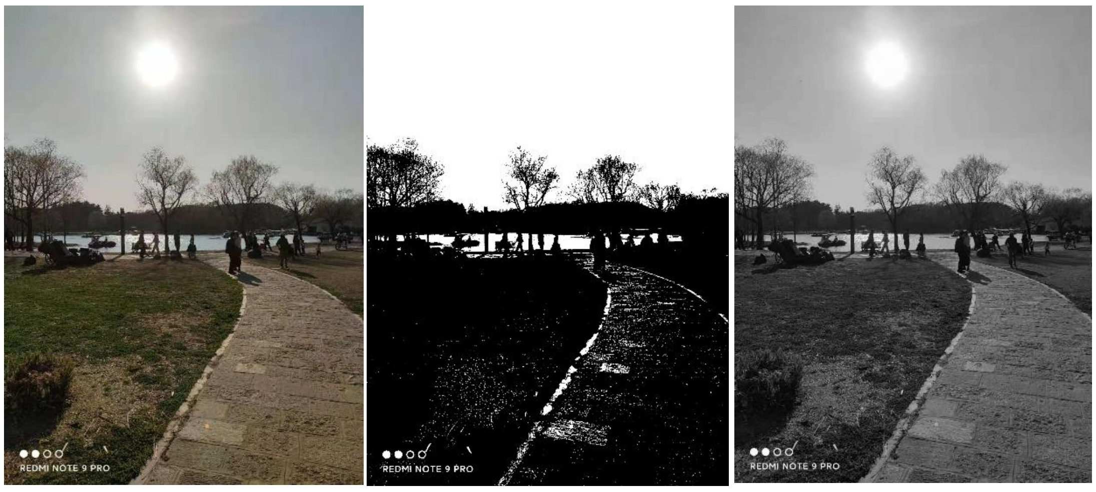
```
Objects
Name      Value
dk        1      Regi...  Internal
RGB       20     Pack...  Internal
out_gray  5      Net      Internal

Wave - Default
/gray_tb/dk        1
/gray_tb/RGB       00000000000000...
/gray_tb/out_gray  00000101

RGB:      00000000000000000... | 0000000000000000000001010 | 000000000000001111101000
out_gray: 00000101 | 00000010 | 00111011

Now       100 ns
Cursor 1  12 ns
```

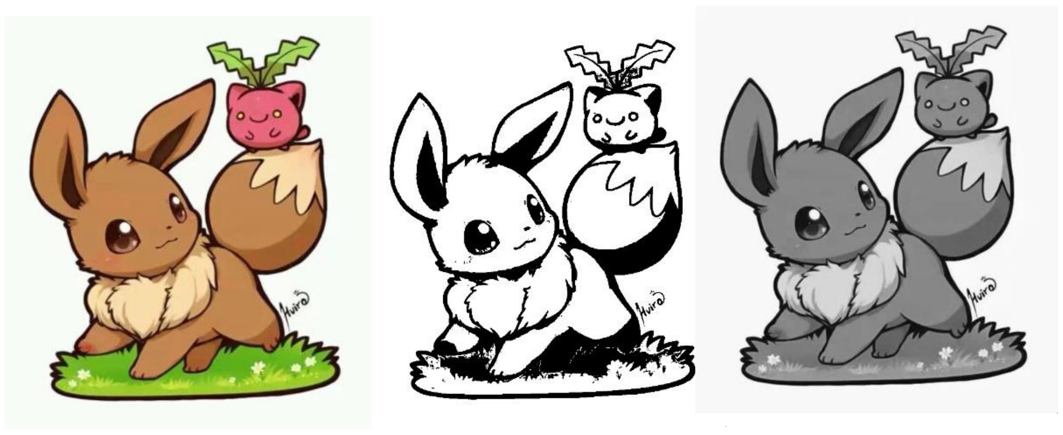
Testbench for grayscale image



```
Objects
Name    Value
dk      0         Regi...  Internal
gray    11111111  Pack...  Internal
binary  11111111  Net      Internal

Wave - Default
/binary_tb/dk      0
/binary_tb/gray    00001010
/binary_tb/binary  00000000

gray:   00001010 | 01100100 | 10000000 | 11001000
binary: 00000000 | 11111111

Now       200 ns
Cursor 1  24 ns
```

Testbench for binarization





Examples for Image Process

Problems and Improvements

1. The deprecated versions of Lab 4

The original version of Lab 4 in the documentation is already deprecated by Windows and Visual Studio. With the original version in hand, I was totally confused with lots of error messages and was then a bit informed when noticing the previous projects in 前人 PJ. After following strictly the previous project, I finally completed the whole process of documentation after turning to the new DLL. And to be specific, in my Lab 4 using OpenCV, it is unfortunate that the latest OpenCV does not contain a x86 version of runtime library and old OpenCV does not get along well with the latest Visual Studio as well, so the change of DLL is a must for my lab.

2. The deprecated VideoPlayer in VeriComm

It should be mentioned that doing image process seems to be simpler using VideoPlayer in VeriComm as shown in documentation. However, it is no longer supported by Windows 10 and will pop up a message telling that several libraries are not found. Looking into the files of the project, we will find processed data but no players are given to run those data.

3. More knowledge necessary for image process and video capture

It is quite fancy to give out an outcome of pictures, but a lack of knowledge of image process is constraining my work. Actually, I once found a previous project using Sobel factor to detect the edge of images, but it uses a BRAM to store 3×3 pixel data which is out of my control of FDE. To make it a bit further, most complicated image process methods need a clutch of pixel data to process at one time so they are almost incompatible with FDE.