**2023 Spring VLSI DSP Homework Assignment #3**

4108064101 杜冠廷

1. **Determining the word length of the filter coefficients**

Matlab code:

|  |
| --- |
| Computing the filter coefficients difference between floating point and fixed point |
| clear all  close all  clc  %Filter coefficients  h = [ 0.037828455507;  -0.023849465020;  -0.110624404418;  0.377402855613;  0.852698679009;  0.377402855613;  -0.110624404418;  -0.023849465020;  0.037828455507];  g = [-0.064538882629;  0.040689417609;  0.418092273222;  -0.788485616406;  0.418092273222;  0.040689417609;  -0.064538882629];  W = 16;  h\_error = zeros(W, 1);  g\_error = zeros(W, 1);  for i = 8 : W  h\_fix = fi(h, 1, i, i-1);  g\_fix = fi(g, 1, i, i-1);  h\_error(i) = sum(abs(h - double(h\_fix))) \* 100;  g\_error(i) = sum(abs(g - double(g\_fix))) \* 100;  end  figure(1)  hold on  plot(h\_error,'k\*-');  plot(g\_error,'bs-');  legend('h', 'g');  xlabel('coefficient word length');  ylabel('Coefficient difference (%)');  xlim([8 W]);  ylim([0 2.0]);  title('Filter coefficients difference');  hold off |

Result:

|  |
| --- |
|  |
| Choosing the word length of the filter coefficients are 11. |

1. **Determining the word length of the filter outputs at each level**

Matlab code:

|  |
| --- |
| clear all  close all  clc  %Read image  test\_image = fi(imread('HW3\_test\_image.bmp') ,0 ,8 ,0);  %Define output result matrix  DWT\_out = zeros(512,512);  %Define image process length  L = 512;  %Define fixed point precision  layer1\_fixed = numerictype(true, 10, 0); % max : 504 min : -223 width(min.) = f + 9 + 1 (f == fraction number)  layer2\_fixed = numerictype(true, 11, 0); % max : 985 min : -295 width(min.) = f + 10 + 1 (f == fraction number)  layer3\_fixed = numerictype(true, 12, 0); % max : 1952 min : -469 width(min.) = f + 11 + 1 (f == fraction number)  %DWT  [out\_l ,out\_h] = DWT\_row\_processing(L ,test\_image ,layer1\_fixed);  [out\_ll ,out\_hl ,out\_lh ,out\_hh] = DWT\_column\_processing(L ,out\_l ,out\_h ,layer1\_fixed);  DWT\_out(1 : 256 ,257 : 512) = out\_hl;  DWT\_out(257 : 512 ,1 : 256) = out\_lh;  DWT\_out(257 : 512 ,257 : 512) = out\_hh;  [out\_l\_2 ,out\_h\_2] = DWT\_row\_processing(256 ,out\_ll ,layer2\_fixed);  [out\_ll\_2 ,out\_hl\_2 ,out\_lh\_2 ,out\_hh\_2] = DWT\_column\_processing(256 ,out\_l\_2 ,out\_h\_2 ,layer2\_fixed);  DWT\_out(1 : 128 ,129 : 256) = out\_hl\_2;  DWT\_out(129 : 256 ,1 : 128) = out\_lh\_2;  DWT\_out(129 : 256 ,129 : 256) = out\_hh\_2;  [out\_l\_3 ,out\_h\_3] = DWT\_row\_processing(128 ,out\_ll\_2 ,layer3\_fixed);  [out\_ll\_3 ,out\_hl\_3 ,out\_lh\_3 ,out\_hh\_3] = DWT\_column\_processing(128 ,out\_l\_3 ,out\_h\_3 ,layer3\_fixed);  DWT\_out(1 : 64 ,1 : 64) = out\_ll\_3;  DWT\_out(1 : 64 ,65 : 128) = out\_hl\_3;  DWT\_out(65 : 128 ,1 : 64) = out\_lh\_3;  DWT\_out(65 : 128 ,65 : 128) = out\_hh\_3;  %IDWT  [I\_out\_l\_2 ,I\_out\_h\_2] = IDWT\_column\_processing(128 ,out\_ll\_3 ,out\_lh\_3 ,out\_hl\_3 ,out\_hh\_3);  [inv\_pic\_2] = IDWT\_row\_processing(128 ,I\_out\_l\_2 ,I\_out\_h\_2);  [I\_out\_l\_1 ,I\_out\_h\_1] = IDWT\_column\_processing(256 ,inv\_pic\_2 ,out\_lh\_2 ,out\_hl\_2 ,out\_hh\_2);  [inv\_pic\_1] = IDWT\_row\_processing(256 ,I\_out\_l\_1 ,I\_out\_h\_1);  [I\_out\_l ,I\_out\_h] = IDWT\_column\_processing(L ,inv\_pic\_1 ,out\_lh ,out\_hl ,out\_hh);  [inv\_pic] = IDWT\_row\_processing(L ,I\_out\_l ,I\_out\_h);  %PSNR  MSE = 0;  for i = 1 : 512  for j = 1 : 512  MSE = MSE + ((double(test\_image(i ,j)) - double(inv\_pic(i ,j))) ^ 2);  end  end  MSE = MSE / (512 ^ 2);  PSNR = 10 \* (log10((255 ^ 2) / MSE));  disp(['PSNR = ',num2str(PSNR) ,' dB']);  %Display image after processing  figure(1)  imshow(mat2gray(double(test\_image)));  figure(2)  imshow(mat2gray(double(inv\_pic))); |
| function [out\_l ,out\_h] = DWT\_row\_processing(L ,pic ,fixed\_size)  %Filter coefficients  h\_floating = [ 0.037828455507;  -0.023849465020;  -0.110624404418;  0.377402855613;  0.852698679009;  0.377402855613;  -0.110624404418;  -0.023849465020;  0.037828455507];  g\_floating = [-0.064538882629;  0.040689417609;  0.418092273222;  -0.788485616406;  0.418092273222;  0.040689417609;  -0.064538882629];  %Fixed point coefficients  h = fi(h\_floating ,1 ,11 ,10);  g = fi(g\_floating ,1 ,11 ,10);  %Symmetric extension at picture boundary  p\_l = zeros(L ,L + 8 ,'like', fi([], fixed\_size));  p\_l = [pic( : ,5) pic( : ,4) pic( : ,3) pic( : ,2) pic pic( : ,L - 1) pic( : ,L - 2) pic( : ,L - 3) pic( : ,L - 4)];  p\_h = zeros(L ,L + 6 ,'like', fi([], fixed\_size));  p\_h = [pic( : ,4) pic( : ,3) pic( : ,2) pic pic( : ,L - 1) pic( : ,L - 2) pic( : ,L - 3)];  %Compute output picture  for i = 1 : L  %Lowpass filter  temp\_l = conv(p\_l(i , :) ,h);  out\_l(i , 1 : (L / 2)) = fi(temp\_l(1 ,9 : 2 : (L + 7)) ,fixed\_size);  %Highpass filter  temp\_h = conv(p\_h(i , :) ,g);  out\_h(i , 1 : (L / 2)) = fi(temp\_h(1 ,8 : 2 : (L + 6)) ,fixed\_size);  end  end |
| function [out\_ll ,out\_hl ,out\_lh ,out\_hh] = DWT\_column\_processing(L ,input\_l ,input\_h ,fixed\_size)  %Filter coefficients  h\_floating = [ 0.037828455507;  -0.023849465020;  -0.110624404418;  0.377402855613;  0.852698679009;  0.377402855613;  -0.110624404418;  -0.023849465020;  0.037828455507];  g\_floating = [-0.064538882629;  0.040689417609;  0.418092273222;  -0.788485616406;  0.418092273222;  0.040689417609;  -0.064538882629];  %Fixed point coefficients  h = fi(h\_floating ,1 ,11 ,10);  g = fi(g\_floating ,1 ,11 ,10);  %Symmetric extension at picture boundary  input\_l\_extension\_for\_l = zeros(L + 8 ,L / 2 ,'like', fi([], fixed\_size));  input\_l\_extension\_for\_l = [input\_l(5 , : ); input\_l(4 , : ); input\_l(3 , : ); input\_l(2 , : ); input\_l; input\_l(L - 1 , : ); input\_l(L - 2 , : ); input\_l(L - 3, : ); input\_l(L - 4, : )];  input\_l\_extension\_for\_h = zeros(L + 6 ,L / 2 ,'like', fi([], fixed\_size));  input\_l\_extension\_for\_h = [input\_l(4 , : ); input\_l(3 , : ); input\_l(2 , : ); input\_l; input\_l(L - 1 , : ); input\_l(L - 2 , : ); input\_l(L - 3, : )];  input\_h\_extension\_for\_l = zeros(L + 8 ,L / 2 ,'like', fi([], fixed\_size));  input\_h\_extension\_for\_l = [input\_h(5 , : ); input\_h(4 , : ); input\_h(3 , : ); input\_h(2 , : ); input\_h; input\_h(L - 1 , : ); input\_h(L - 2 , : ); input\_h(L - 3, : ); input\_h(L - 4, : )];  input\_h\_extension\_for\_h = zeros(L + 6 ,L / 2 ,'like', fi([], fixed\_size));  input\_h\_extension\_for\_h = [input\_h(4 , : ); input\_h(3 , : ); input\_h(2 , : ); input\_h; input\_h(L - 1 , : ); input\_h(L - 2 , : ); input\_h(L - 3, : )];  %Compute output picture  for i = 1 : L/2  %Lowpass filter  temp\_ll = conv(input\_l\_extension\_for\_l( : ,i) ,h);  out\_ll(1 : (L / 2) ,i) = fi(temp\_ll(9 : 2 : (L + 7) ,1) ,fixed\_size);  temp\_hl = conv(input\_h\_extension\_for\_l( : ,i) ,h);  out\_hl(1 : (L / 2) ,i) = fi(temp\_hl(9 : 2 : (L + 7) ,1) ,fixed\_size);  %Highpass filter  temp\_lh = conv(input\_l\_extension\_for\_h( : ,i) ,g);  out\_lh(1 : (L / 2) ,i) = fi(temp\_lh(8 : 2 : (L + 6) ,1) ,fixed\_size);  temp\_hh = conv(input\_h\_extension\_for\_h( : ,i) ,g);  out\_hh(1 : (L / 2) ,i) = fi(temp\_hh(8 : 2 : (L + 6) ,1) ,fixed\_size);  end  end |
| IDWT\_column\_processing function and IDWT\_row\_processing function are the same as HW2 assignment. |

Result:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| The word length of the filter outputs at layer 1:   |  |  | | --- | --- | | Data Type Mode | Fixed-point: binary point scaling | | Signed or unsigned | Signed | | Word Length | 10 | | Sign bit Length | 1 | | Integer Length | 9 | | Fraction Length | 0 |   The word length of the filter outputs at layer 2:   |  |  | | --- | --- | | Data Type Mode | Fixed-point: binary point scaling | | Signed or unsigned | Signed | | Word Length | 11 | | Sign bit Length | 1 | | Integer Length | 10 | | Fraction Length | 0 |   The word length of the filter outputs at layer 3:   |  |  | | --- | --- | | Data Type Mode | Fixed-point: binary point scaling | | Signed or unsigned | Signed | | Word Length | 12 | | Sign bit Length | 1 | | Integer Length | 11 | | Fraction Length | 0 | |

1. **The area of each multiplier, adder, and register**

Verilog code:

|  |
| --- |
| Multiplier |
|  |
| Adder |
|  |
| Register |
|  |

Result:

|  |  |  |
| --- | --- | --- |
| Multiplier | | |
| Layer 1 |  | |
| Layer 2 |  | |
| Layer 3 |  | |
| Adder | | |
| Layer 1 |  | |
| Layer 2 |  | |
| Layer 3 |  | |
| Register | | |
| 1 bit register | |  |
| Layer 1  output | |  |
| Layer 2  output | |  |
| Layer 3  output | |  |

Note:

|  |  |
| --- | --- |
| Cell Library | Cell-Based Design Kit for IC Contest  製程: TSMC 0.13um  廠商: ARM  Design Kit版本：CBDK\_IC\_Contest\_v2.5 |
| Synthesis tool | Design Compiler  廠商: Synopsys |

***Acknowledgment***

NCHU EE Undergraduate Chun-Wei Su

NCHU EE ICs & Systems Lab 612 Shun-Liang Yeh