Introduction

The digital video scalers in the Zipcores IP library use a series of polyphase filters in the x and y dimensions. This application note addresses some of the possible methods for sampling coefficient sets for these types of filter.

The application note begins with a brief description of how a polyphase filter works and uses a simple Bi-linear filter example as a starting point. The discussion continues with a description of a 5-tap polyphase filter and describes how to generate coefficients using a Lanczos2-windowed sinc function. Finally, the application note provides an example Matlab® script for the automatic generation of coefficient sets.

Polyphase filter architecture

Each 'phase' of a digital polyphase filter has it's own unique set of coefficients which represent the filter impulse response for that particular phase. In the context of the Zipcores video scaler filters, the phase refers to the interpolation point used between pixels in the source image. This means that as the interpolation point between pixels changes, then so does the phase and the resulting filter coefficients.

Mathematically, an 'n' phase filter could be designed as 'n' discrete filters in parallel - but obviously this would not make for a very practical design. In practice, the Zipcores video filters are designed as a single filter with multiplexed coefficients controlled by the phase. Figure 1 shows an example architecture of an 'n' phase, 'm' tap polyphase filter.

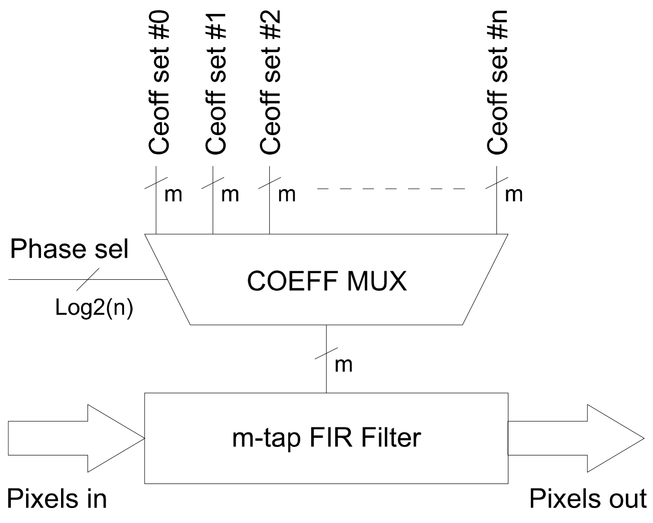

Figure 1: N-Phase, M-tap polyphase filter Architecture

Generating coefficient sets

For each phase of a polyphase filter, the filter impulse response changes. This means that a different set of coefficients is required for each phase. The easiest approach is to start by considering the characteristic of a filter with only 2 taps. A 2-tap filter employs linear interpolation between adjacent pixels. The following paragraphs demonstrate a number of techniques for generating coefficient sets.

Bilinear polyphase filter

For a bilinear polyphase filter, the tap positions for phase '0' are located as described by Figure 2.

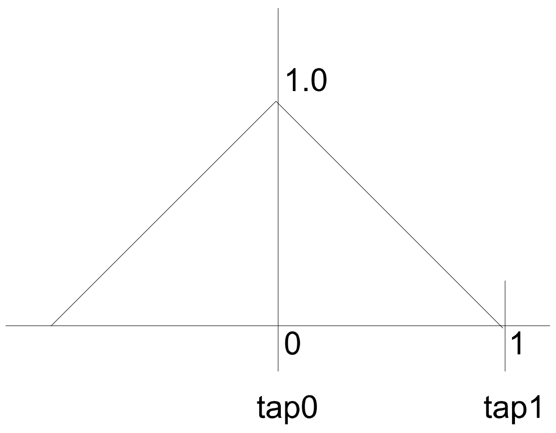

Figure 2: Bi-linear filter tap positioning

As the interpolation point between pixels moves in the source image, so does the phase. Zipcores filters generally have 16 phases meaning that as the phase changes, the filter characteristic also changes in a way described by Figure 3. As the phase increments, the curve is shifted to the right by 1/16. Filter coefficients are sampled for each phase at the points where the curve intersects the y-axis for taps 0 and 1. Figure 3 shows a simple linear interpolation between pixels.

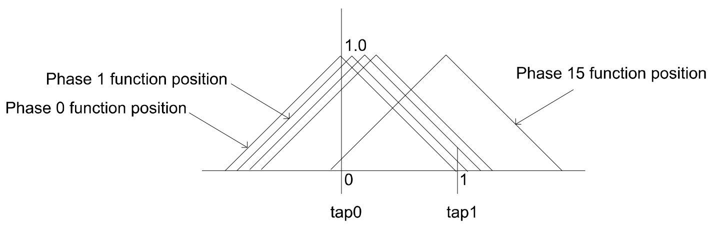
1.0
Phase 1 function position
Phase 0 function position
Phase 15 function position
0
1
tap0
tap1

Figure 3: Sampling filter coefficients over 16 phases

Polyphase filters with more taps

For larger numbers of taps, a common approach is to use a windowed-sinc function to design the filter kernel. To start with, the centre tap of the filter is positioned over the central lobe of the function. Figure 5 shows an example of this for a lanczos2-windowed sinc.

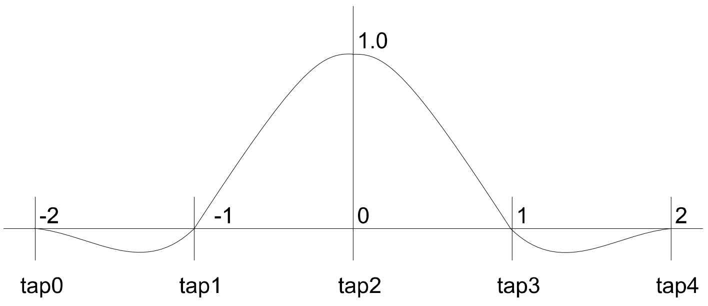

Figure 4: Filter tap positioning for the Lanczos2-windowed sinc

In this example, tap2 is the centre tap. In order to derive coefficients for each phase of a 16 phase filter, the principal is exactly the same as for the bi-linear case. As the phase increments, the curve is shifted to the right in steps of 1/16. The filter coefficients are sampled for taps 0,1,2,3 and 4 where the curve intersects the y-axis. Figure 6 again demonstrates this technique pictorially.

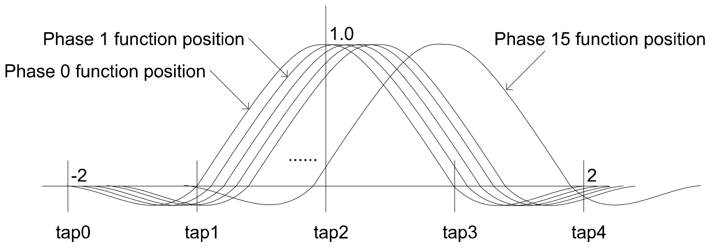
Phase 1 function position
Phase 0 function position
1.0
Phase 15 function position
......
-2
2
tap0
tap1
tap2
tap3
tap4

Figure 5: Sampling filter coefficients over 16 phases (5 taps)

It is important to note that once the coefficients have been sampled and quantized, the sum of each set of coefficients (the DC gain) should sum to unity - otherwise visible artefacts may be present in the scaled output image.

Choice of windowed sinc function

The best windowed sinc function to use for video scaling applications is a matter of personal preference. The best method is to experiment with different functions for your given application. Results will vary depending on the scale factor and the type of image being scaled. Generally the Lanczos2-windowed sinc gives good all round performance and is commonly used for video scaling applications. Other popular windows used are Hamming, Kaiser and Blackman. Figure 7 shows these example filter windows sampled over the interval -2,2.

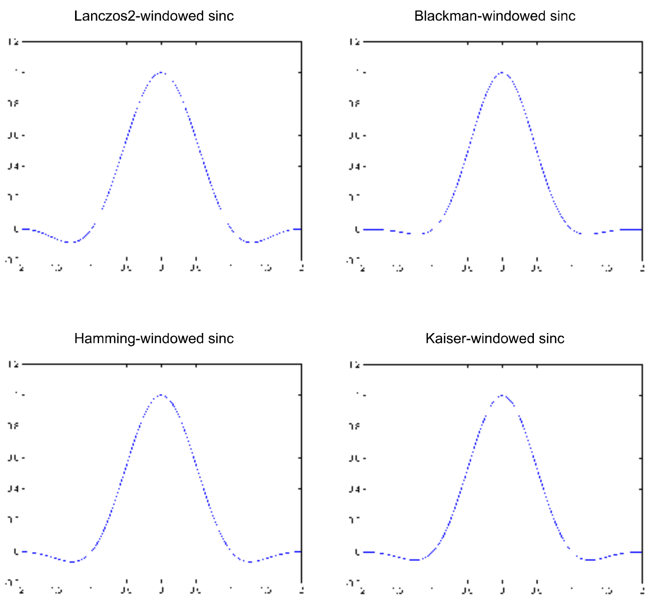

Figure 6: Some common windowed-sinc functions

The functions in Figure 7 were generated in Matab® using the following short script. The number of filter taps may be adjusted by modifying the value 'N'. Currently this is set for a 5 tap filter.

```
% Lanczos2-windowed sinc over range -N/2,N/2
N  = 5;
ks = -(N-1)/2:0.001:(N-1)/2;
kw = -(N-1)/2:0.001:(N-1)/2;

s  = sinc(ks);
w  = sinc(kw/2);

ws = s.*w;
plot (ks,ws)


% Blackman-windowed sinc over range -N/2,N/2
N  = 5;
ks = -(N-1)/2:0.001:(N-1)/2;
kw = 0:0.001:(N-1);

s  = sinc(ks);
w  = 0.42 - 0.5*cos(2*pi*kw/(N-1)) +
0.08*cos(4*pi*kw/(N-1));

ws = s.*w;
plot (ks,ws)


% Hamming-windowed sinc over range -N/2,N/2
N  = 5;
ks = -(N-1)/2:0.001:(N-1)/2;
kw = 0:0.001:(N-1);

s  = sinc(ks);
w  = 0.53836 - 0.46164*cos(2*pi*kw/(N-1));

ws = s.*w;
plot (ks,ws)
```

```
% Kaiser-windowed sinc over range -N/2,N/2
N  = 5;
ks = -(N-1)/2:0.001:(N-1)/2;
kw = 0:0.001:(N-1);

s  = sinc(ks);
alpha = 2*pi;
w  = besseli(0,alpha*sqrt(1-(2*kw/(N-1) -1).^2))
/besseli(0,alpha);

ws = s.*w;
plot (ks,ws)
```

Generating Coefficients in Matlab®

The next script samples the coefficients for the Lanczos2-windowed sinc function for 5 taps and 16 phases as per the description in Figure 6. The script also generates a text file output.

```
% Calculate coefficients for Phases 0 to 15,
% Taps 0,1,2,3,4

for p_index = 1:16
    for t_index = 1:5
        p = (p_index - 1)/16;
        t = (t_index - 1);
        x = t - 2 - p;
        coeff(p_index, t_index) = sinc(x) * sinc(x/2);
    end
end


% Quantize to 2.6-bit signed numbers
coeff_quant = round(coeff * 64);


% Check they sum to 1
sum_coeff       = sum(coeff,2);
sum_coeff_quant = sum(coeff_quant,2);


% Write coefficients to a file I=Phase, J=Tap
fid = fopen('coeffs_5tap.txt', 'w');
fprintf(fid, '            TAP0  TAP1  TAP2  TAP3  TAP4
SUM\n\n');
for I = 1:16
    fprintf(fid, 'PHASE%2d : ',I-1 );
    for J = 1:5
        fprintf(fid, '%5d ', coeff_quant(I,J));
    end

    if sum_coeff_quant(I) == 64
        fprintf(fid, '%5d\n', sum_coeff_quant(I));
    else
        fprintf(fid, '%5d *\n', sum_coeff_quant(I));
    end
end

fclose(fid);
```

Coefficient file output

Note that minor adjustments to the coefficients must be made in order that they sum to unity. These values are labelled with an asterisk '*' in the output text file. All coefficients are output as 2.6-bit signed values.

```
            TAP0  TAP1  TAP2  TAP3  TAP4   SUM

PHASE 0 :      0     0    64     0     0    64
PHASE 1 :      0    -2    63     3     0    64
PHASE 2 :      0    -4    62     6     0    64
PHASE 3 :      0    -5    59    10    -1    63 *
PHASE 4 :     -1    -5    56    15    -1    64
PHASE 5 :     -1    -6    52    20    -2    63 *
PHASE 6 :     -1    -5    47    26    -3    64
PHASE 7 :     -1    -5    42    31    -3    64
PHASE 8 :     -1    -4    37    37    -4    65 *
PHASE 9 :     -1    -3    31    42    -5    64
PHASE10 :     -1    -3    26    47    -5    64
PHASE11 :     -1    -2    20    52    -6    63 *
PHASE12 :     -1    -1    15    56    -5    64
PHASE13 :     -1    -1    10    59    -5    62 *
PHASE14 :     -1     0     6    62    -4    63 *
PHASE15 :      0     0     3    63    -2    64
```

Conclusion

In this application note, a detailed explanation of polyphase filters has been provided in the context of digital video scaling applications. A number of techniques have been presented for generating sets of coefficients for the filter phases and, in addition, the application note has described some common windowed-sinc functions for deriving the filter kernels.

Finally, a Matlab® script has been provided to automatically generate coefficients for a 5-tap, 16-phase filter using a Lanczos2-windowed sinc function.

Revision History

Revision	***Change description***	***Date***
1.0	Initial revision	06/03/2009
1.1	Removed superfluous information and simplified the text	21/01/2013