EE24MTECH14022 Assignment 5

1 What is an FIR Filter?

A **Finite Impulse Response (FIR) filter** is a type of digital filter where the impulse response settles to zero after a finite number of samples. FIR filters are widely used in signal processing because they are always stable and can be designed to have a linear phase response.

2 General FIR Filter Equation

The output of an FIR filter is given by:

$$y[n] = \sum_{k=0}^{N-1} h[k]x[n-k] \tag{1}$$

where:

- $x[n]$ is the input signal,
- $h[k]$ are the filter coefficients (impulse response),
- N is the number of filter coefficients (or taps),
- $y[n]$ is the output signal.

3 Matlab Code for Signal Generation and Filter Application

```
% Load filter coefficients
coeffs = load('bandpass_filter2.mat');  % Load the file
    containing filter coefficients
b = coeffs.Num;  % Assuming the coefficients are stored in a
     variable named 'Num'

% Extract the first 20 coefficients
b = b(1:20);

```

```
8
9  function fixed_out = to_fixed(x)
10     Q8_24_factor = 2^24;
11     fixed_out = round(x * Q8_24_factor);
12     fixed_out = max(min(fixed_out,2^31 - 1), -2^31);
13 end
14
15 function fixed_mul = mul(x,y)
16     Q8_24_factor = 2^24;
17     fixed_mul = round((x * y) / Q8_24_factor);
18 end
19
20 function fir_out = fir(x, h)
21     % Initialize the buffer to store previous input samples
22     persistent buffer;
23
24     % Initialize the buffer on the first run
25     if isempty(buffer)
26         buffer = zeros(1, length(h));  % Buffer size equals
               the number of filter coefficients
27     end
28
29     % Shift the buffer: move values in the buffer to the
           right
30     buffer(2:end) = buffer(1:end-1);  % Shift all elements
           one position to the right
31
32     % Insert the new sample at the beginning of the buffer
33     buffer(1) = x;
34
35     % Perform the convolution: multiply each buffered sample
            by the corresponding filter coefficient
36     fir_out = 0;
37     for i = 1:length(h)
38         fir_out = fir_out + mul(buffer(i) ,h(i));  % Sum of
               products (FIR convolution)
39     end
40 end
41
42
43 b_fixed = to_fixed(b);
44
45 % Save filter coefficients
46 fid = fopen('filter_coeffs_fixed.txt', 'w');
47 fprintf(fid, '%d\n', b_fixed);
48 fclose(fid);
49
50
51
52 % Generate and filter sine waves
```

```
fs = 48000;  % Sampling frequency
cycles = 5;  % Number of cycles
freqs = [100, 2000, 6000, 11000];

% Use a fixed time vector (same samples for all signals)
T_min = 1 / min(freqs);
t_fixed = 0:1/fs:(cycles * T_min) - 1/fs;
Q8_24_factor = 2^24;
figure;
for i = 1:length(freqs)
    sinewave = sin(2 * pi * freqs(i) * t_fixed);  % Generate sine wave

    % Apply FIR filter (convolution)


    % Convert sine wave to Q(2,14)
    sine_fixed = round(sinewave * Q8_24_factor);
    sine_fixed = max(min(sine_fixed, 2^31-1), -2^31);  % Saturate


    % Save original sine wave
    filename = sprintf('sinewave_%dHz_fixed.txt', freqs(i));
    fid = fopen(filename, 'w');
    fprintf(fid, '%d\n', sine_fixed);
    fclose(fid);

    N = length(sine_fixed);
    filtered_signal = zeros(1,N);
    for j = 1:N
        filtered_signal(j) = fir(sine_fixed(j),b_fixed);
    end
    filtered_signal_float = filtered_signal/Q8_24_factor;


    % % Save filtered signal
    % filename = sprintf('filtered_sinewave_%dHz_fixed.txt', freqs(i));
    % fid = fopen(filename, 'w');
    % fprintf(fid, '%d\n', filtered_fixed_twos);
    % fclose(fid);

    % Plot original and filtered signals
    subplot(length(freqs), 2, 2*i-1);
    plot(t_fixed, sinewave, 'b', 'LineWidth', 1.5);
    title(sprintf('Original Signal - %d Hz', freqs(i)));
    xlabel('Time (s)');
    ylabel('Amplitude');
    grid on;
```

```
100
101     subplot(length(freqs), 2, 2*i);
102     plot(t_fixed, filtered_signal_float, 'r', 'LineWidth',
            1.5);
103     title(sprintf('Filtered Signal - %d Hz', freqs(i)));
104     xlabel('Time (s)');
105     ylabel('Amplitude');
106     grid on;
107 end
108
109 fid = fopen('output_100.txt','r');
110 data = fscanf(fid,'%d');
111 fclose(fid);
112
113 figure;
114 subplot(4,1,1);
115 plot(t_fixed, data/2^24, 'b-', 'LineWidth', 1.5);
116 title('verilog Filtered Signal - 100Hz');
117     xlabel('Time (s)');
118     ylabel('Amplitude');
119 grid on;
120
121
122 subplot(4,1,2);
123
124 fid = fopen('output_2000.txt');
125 data = fscanf(fid,'%d');
126 fclose(fid);
127 plot(t_fixed,data/2^24, 'b-', 'LineWidth', 1.5);
128 title('verilog Filtered Signal - 2000Hz');
129     xlabel('Time (s)');
130     ylabel('Amplitude');
131 grid on;
132
133 disp('All files and plots for filtered sine waves have been
        generated successfully.');
134 % Load filter coefficients
135
136 subplot(4,1,3);
137
138 fid = fopen('output_6000.txt');
139 data = fscanf(fid,'%d');
140 fclose(fid);
141 plot(t_fixed,data/2^24, 'b-', 'LineWidth', 1.5);
142 title('verilog Filtered Signal - 6000Hz');
143     xlabel('Time (s)');
144     ylabel('Amplitude');
145 grid on;
146
```

```
disp('All files and plots for filtered sine waves have been
    generated successfully.');

subplot(4,1,4);

fid = fopen('output_11000.txt');
data = fscanf(fid,'%d');
fclose(fid);
plot(t_fixed,data/2^24, 'b-', 'LineWidth', 1.5);
title('verilog Filtered Signal - 11000Hz');
    xlabel('Time (s)');
    ylabel('Amplitude');
grid on;

disp('All files and plots for filtered sine waves have been
    generated successfully.');
```

Listing 1: Matlab code for Signal Genration and Visualization

4 Verilog Code

```
module fir(
    input wire clk,           // Clock signal
    input wire rst_n,         // Active low reset
    input signed [31:0] x,      // Input sample (Q8.24
        format)
    output reg signed [31:0] y       // Output filtered
        sample (Q8.24 format)
);
    // Define filter coefficients (b)
    reg signed [31:0] coeffs [0:19];   // 20 filter
        coefficients (Q8.24 format)

    // Shift registers for input samples (buffer)
    reg signed [31:0] buffer [0:19];   // Shift register
        buffer

    // Internal accumulator for FIR output
    reg signed [63:0] accumulator;     // Accumulator (Q8.24
         * Q8.24 gives Q16.48)

    // Load filter coefficients from file
    integer file, i, status;
    initial begin
       coeffs[0] <= -4894;
                 coeffs[1] <= -10754;
                 coeffs[2] <= -9052;
                 coeffs[3] <= 13025;
```

```
                    coeffs[4]  <= 58336;
                    coeffs[5]  <= 107984;
                    coeffs[6]  <= 125575;
                    coeffs[7]  <= 82259;
                    coeffs[8]  <= -13974;
                    coeffs[9]  <= -113010;
                    coeffs[10] <= -154735;
                    coeffs[11] <= -117770;
                    coeffs[12] <= -41762;
                    coeffs[13] <= 3039;
                    coeffs[14] <= -22379;
                    coeffs[15] <= -88604;
                    coeffs[16] <= -124388;
                    coeffs[17] <= -86346;
                    coeffs[18] <= -4717;
                    coeffs[19] <= 43988;
    end

    // Update the buffer and accumulator on each clock cycle
    always @(posedge clk or negedge rst_n) begin
        if (~rst_n) begin
            // Reset buffer and accumulator on reset
            accumulator = 64'd0;
            for (i = 0; i < 20; i = i + 1) begin
                buffer[i] = 32'd0;
            end
        end else begin
            // Shift the buffer (delay previous samples)
            for (i = 19; i > 0; i = i - 1) begin
                buffer[i] = buffer[i - 1];
            end
            buffer[0] = x;  // Insert new input sample at
                the front of the buffer

            // Reset the accumulator
            accumulator = 64'd0;

            // Perform the FIR convolution (multiply-and-
                accumulate)
            for (i = 0; i < 20; i = i + 1) begin
                accumulator = accumulator + (buffer[i] *
                    coeffs[i]);
            end

            // The output is the accumulator value (scaled
                back to Q8.24 format)
            y = accumulator >> 24;
    end
    end
endmodule
```

Listing 2: Verilog FIR non-pipeline code

```
module fir_tb;
    reg clk;
    reg rst_n;
    reg signed [31:0] x;        // Input sample (Q8.24 format)
    wire signed [31:0] y;       // Output filtered sample

    // Instantiate the FIR filter module
    fir_filter uut (
        .clk(clk),
        .rst_n(rst_n),
        .x(x),
        .y(y)
    );

    // Clock generation
    always begin
        #5 clk = ~clk;  // Clock period of 10ns
    end

    // Memory array to hold 2400 input values
    reg signed [31:0] input_mem [0:2399]; // Array to hold 2400 input samples
    integer i;
    integer input_file,num_samples;

    // Read the input file into memory
    initial begin
        // Initialize signals
        clk = 0;
        rst_n = 0;
        x = 32'd0;

        // Apply reset
        #10 rst_n = 1;

        // Open the input file (decimal format)
        input_file = $fopen("sinewave_2000Hz_fixed.txt", "r");
        if (input_file == 0) begin
            $display("Error: Unable to open input file.");
            $finish;
        end

        num_samples = 0;
        while (!$feof(input_file) && num_samples < 2400)
```

```
            begin
44              if ($fscanf(input_file, "%d\n", input_mem[
                    num_samples]) == 1)
45                  num_samples = num_samples + 1;
46
47          end
48
49          // Close the file after reading
50          $fclose(input_file);
51      end
52      initial begin
53
54          // Apply input samples to the filter
55          for (i = 0; i < 2400; i = i + 1) begin
56              #10 x = input_mem[i];  // Provide input sample
                    to the filter
57                  $display("%d", y);
58          end
59
60                              $finish;
61
62      end
63
64
65
66
67      // Monitor the output
68      initial begin
69          $monitor("At time %t, input = %d, output = %d,
                num_samples= %d", $time, x, y,num_samples);
70      end
71  endmodule
```

Listing 3: Verilog FIR non-pipelining testbench code

```
1
2
3   // Experiment 5:
4
5   // Mannava Venkatasai
6   // EE24MTECH12008
7   // FIR filter Design using pipe lining
8   // input is in Q14 format
9
10
11  `timescale 1ns/1ps
12  module fir_filter(
13      input wire clk,
14      input rst,
15      input signed [15:0] input_signal,
```

```
    output reg signed[15:0] output_signal
);

parameter N = 123;

reg signed [15:0]filter_coefcients[0:N-1];
reg signed [15:0]shift_reg[0:N-1];
reg signed [15:0]shift_reg_temp[0:N-1];
reg signed [31:0]mul_reg[0:N-1];


integer i;
integer k;
integer count;
reg [31:0]d_out;
reg [31:0]acc;
reg [15:0]temp;

initial
    begin

        filter_coefcients[0] = -5;
        filter_coefcients[1] = -11;
        filter_coefcients[2] = -9;
        filter_coefcients[3] = 13;
        filter_coefcients[4] = 57;
        filter_coefcients[5] = 105;
        filter_coefcients[6] = 123;
        filter_coefcients[7] = 80;
        filter_coefcients[8] = -14;
        filter_coefcients[9] = -110;
        filter_coefcients[10] = -151;
        filter_coefcients[11] = -115;
        filter_coefcients[12] = -41;
        filter_coefcients[13] = 3;
        filter_coefcients[14] = -22;
        filter_coefcients[15] = -87;
        filter_coefcients[16] = -121;
        filter_coefcients[17] = -84;
        filter_coefcients[18] = -5;
        filter_coefcients[19] = 43;
        filter_coefcients[20] = 15;
        filter_coefcients[21] = -53;
        filter_coefcients[22] = -77;
        filter_coefcients[23] = -13;
        filter_coefcients[24] = 91;
        filter_coefcients[25] = 140;
        filter_coefcients[26] = 90;
        filter_coefcients[27] = 3;
        filter_coefcients[28] = -11;
```

```
66          filter_coefcients[29] = 87;
67          filter_coefcients[30] = 213;
68          filter_coefcients[31] = 242;
69          filter_coefcients[32] = 141;
70          filter_coefcients[33] = 15;
71          filter_coefcients[34] = 6;
72          filter_coefcients[35] = 137;
73          filter_coefcients[36] = 273;
74          filter_coefcients[37] = 249;
75          filter_coefcients[38] = 59;
76          filter_coefcients[39] = -124;
77          filter_coefcients[40] = -119;
78          filter_coefcients[41] = 63;
79          filter_coefcients[42] = 201;
80          filter_coefcients[43] = 90;
81          filter_coefcients[44] = -227;
82          filter_coefcients[45] = -469;
83          filter_coefcients[46] = -404;
84          filter_coefcients[47] = -114;
85          filter_coefcients[48] = 42;
86          filter_coefcients[49] = -204;
87          filter_coefcients[50] = -710;
88          filter_coefcients[51] = -1002;
89          filter_coefcients[52] = -751;
90          filter_coefcients[53] = -180;
91          filter_coefcients[54] = 58;
92          filter_coefcients[55] = -493;
93          filter_coefcients[56] = -1507;
94          filter_coefcients[57] = -1972;
95          filter_coefcients[58] = -991;
96          filter_coefcients[59] = 1341;
97          filter_coefcients[60] = 3799;
98          filter_coefcients[61] = 4849;
99          filter_coefcients[62] = 3799;
100         filter_coefcients[63] = 1341;
101         filter_coefcients[64] = -991;
102         filter_coefcients[65] = -1972;
103         filter_coefcients[66] = -1507;
104         filter_coefcients[67] = -493;
105         filter_coefcients[68] = 58;
106         filter_coefcients[69] = -180;
107         filter_coefcients[70] = -751;
108         filter_coefcients[71] = -1002;
109         filter_coefcients[72] = -710;
110         filter_coefcients[73] = -204;
111         filter_coefcients[74] = 42;
112         filter_coefcients[75] = -114;
113         filter_coefcients[76] = -404;
114         filter_coefcients[77] = -469;
115         filter_coefcients[78] = -227;
```

```
116         filter_coefcients[79] = 90;
117         filter_coefcients[80] = 201;
118         filter_coefcients[81] = 63;
119         filter_coefcients[82] = -119;
120         filter_coefcients[83] = -124;
121         filter_coefcients[84] = 59;
122         filter_coefcients[85] = 249;
123         filter_coefcients[86] = 273;
124         filter_coefcients[87] = 137;
125         filter_coefcients[88] = 6;
126         filter_coefcients[89] = 15;
127         filter_coefcients[90] = 141;
128         filter_coefcients[91] = 242;
129         filter_coefcients[92] = 213;
130         filter_coefcients[93] = 87;
131         filter_coefcients[94] = -11;
132         filter_coefcients[95] = 3;
133         filter_coefcients[96] = 90;
134         filter_coefcients[97] = 140;
135         filter_coefcients[98] = 91;
136         filter_coefcients[99] = -13;
137         filter_coefcients[100] = -77;
138         filter_coefcients[101] = -53;
139         filter_coefcients[102] = 15;
140         filter_coefcients[103] = 43;
141         filter_coefcients[104] = -5;
142         filter_coefcients[105] = -84;
143         filter_coefcients[106] = -121;
144         filter_coefcients[107] = -87;
145         filter_coefcients[108] = -22;
146         filter_coefcients[109] = 3;
147         filter_coefcients[110] = -41;
148         filter_coefcients[111] = -115;
149         filter_coefcients[112] = -151;
150         filter_coefcients[113] = -110;
151         filter_coefcients[114] = -14;
152         filter_coefcients[115] = 80;
153         filter_coefcients[116] = 123;
154         filter_coefcients[117] = 105;
155         filter_coefcients[118] = 57;
156         filter_coefcients[119] = 13;
157         filter_coefcients[120] = -9;
158         filter_coefcients[121] = -11;
159         filter_coefcients[122] = -5;
160
161         for(i=0;i<N;i=i+1)
162         begin
163             shift_reg[i] = 0;
164         end
165         count = 0;
```

```
166     end
167
168     always @ (posedge clk)
169     begin
170         if(rst)
171         begin
172
173             d_out = 0;
174
175         end
176         else
177         begin
178
179         for(i = N-1;i>0;i=i-1)
180         begin
181             shift_reg[i] <= shift_reg[i-1];
182         end
183         shift_reg_temp[0] <= shift_reg[0];
184         if(count < N)
185         begin
186         shift_reg[0] <= input_signal;
187         count <= count + 16'd1;
188         end
189         else
190         begin
191              shift_reg[0] <= 16'b0;
192         end
193
194         // $display("input_signal = %d",input_signal);
195         // $display("time = %t rst = %d shift_reg = [%d%d%d%
                d%d]", $time ,shift_reg[0], shift_reg[1],
                shift_reg[2], shift_reg[3], shift_reg[4],rst);
196
197         acc = 0;
198         for(i = 0;i<N;i=i+1)
199         begin
200             // $display("time = %t shift_reg[%d] = %d ,acc =
                    %d",$time,i,shift_reg[i],acc);
201             mul_reg[i]  <= (shift_reg[i] * filter_coefcients
                    [i])>>14;
202         end
203
204         for(i = 0;i<N;i=i+1)
205         begin
206             // $display("time = %t shift_reg[%d] = %d ,acc =
                    %d",$time,i,shift_reg[i],acc);
207             acc  = acc + mul_reg[i];
208         end
209
210         output_signal <= acc;
```

```
211
212
213         end
214
215
216
217     end
218
219
220
221
222
223 endmodule
```

Listing 4: Verilog FIR pipelined testbench

```
1
2  `timescale 1ns/1ps
3  module fir_filter_tb;
4
5  reg clk;
6  reg rst;
7  reg signed [15:0] x;
8  wire signed [15:0] y;
9  integer i;
10 integer f1,read_status;
11
12 reg signed [15:0] f_input;
13
14 parameter N = 123;
15
16 fir_filter filter(
17     .clk(clk),
18     .rst(rst),
19     .input_signal(x),
20     .output_signal(y)
21 );
22
23 initial
24 begin
25     clk = 0;
26     forever #5 clk = ~clk;
27
28 end
29
30 initial begin
31     $dumpfile("wave.vcd");
32     $dumpvars(0, fir_filter_tb);
33     f1 = $fopen("input_signal_4.txt", "r");
34
```

```
35      rst = 1;
36      #15;
37      rst = 0;
38
39      for (i = 1; i <= 1308 + N -1 + 3; i++)
40      begin
41          read_status = $fscanf(f1, "%d\n", f_input);
42          x = f_input;
43          // $display("\ntime = %t input = %d, rst = %d, out =
                %d", $time, x, rst, y);
44          $display("%d",y);
45          #10;
46      end
47
48      #10;
49      $finish;
50  end
51
52  endmodule
```

Listing 5: Verilog FIR pipelined code

5 Results

Slow 1100mV 85C Model Fmax Summary

<<Filter>>

	Fmax	Restricted Fmax	Clock Name	Note
1	74.4 MHz	74.4 MHz	clk	

Figure 1: FMax Report non-pipelined

Analysis & Synthesis Resource Usage Summary

<<Filter>>

	Resource	Usage
1	Estimate of Logic utilization (ALMs needed)	602
2		
3	Combinational ALUT usage for logic	1000
1	-- 7 input functions	0
2	-- 6 input functions	0
3	-- 5 input functions	0
4	-- 4 input functions	0
5	-- <=3 input functions	1000
4		
5	Dedicated logic registers	640
6		
7	I/O pins	66
8		
9	Total DSP Blocks	60
10		
11	Maximum fan-out node	clk~input
12	Maximum fan-out	640
13	Total fan-out	7068
14	Average fan-out	3.86

Figure 2: Resource utilization non-pipelined

Slow 1100mV 85C Model Fmax Summary

<<Filter>>

	Fmax	Restricted Fmax	Clock Name	Note
1	66.44 MHz	66.44 MHz	clk	

Figure 3: FMax Report pipelined

Analysis & Synthesis Resource Usage Summary

<<Filter>>

	Resource	Usage
1	Estimate of Logic utilization (ALMs needed)	602
2		
3	▾ Combinational ALUT usage for logic	1000
1	-- 7 input functions	0
2	-- 6 input functions	0
3	-- 5 input functions	0
4	-- 4 input functions	0
5	-- <=3 input functions	1000
4		
5	Dedicated logic registers	640
6		
7	I/O pins	66
8		
9	Total DSP Blocks	60
10		
11	Maximum fan-out node	clk~input
12	Maximum fan-out	640
13	Total fan-out	7068
14	Average fan-out	3.86

Figure 4: Resource utilization pipelined