MECHTRON 3TB4 LAB 3 Pre-Lab

Name: Qiushuang Li
Student#: 1206650

1. # of samples = 8000Hz * 50ms * 10^-3 = 400

2. 3-to-1 Multiplexer Verilog HDL Code

```
module dsp_subsystem (input sample_clock,  input reset, input [1:0] selector, input [15:0]
input_sample, output[15:0] output_sample);

wire [15:0] fir;
wire [15:0]     echo;

echo ec (.clk(sample_clock), .reset(reset),.input_signal(input_sample), .output_signal(echo));
assign output_sample = (selector == 2'b00)? fir :
                                                (selector == 2'b01)? echo:
                                                input_sample;

FIR #(24) test (.clk(sample_clock),.reset(reset),.input_signal(input_sample),.output_signal(fir));
endmodule
```

3. FIR filter Verilog HDL Code

```
module FIR (input clk, input reset, input wire[15:0]input_signal, output reg[15:0]output_signal);

parameter N=65;
reg signed[15:0] coeffs[23:0];
reg [15:0] hold[23:0];
wire [15:0] Add[23:0];
genvar i;

always@(*)
begin
         coeffs[0]=-1294;
         coeffs[1]=0;
         coeffs[2]=1482;
         coeffs[3]=0;
        coeffs[4]=-2120;
        coeffs[5]=0;
        coeffs[6]=2741;
        coeffs[7]=0;
        coeffs[8]=-3265;
        coeffs[9]=1;
        coeffs[10]=3613;
        coeffs[11]=1;
        coeffs[12]=29032;
         coeffs[13]=1;
         coeffs[14]=3613;
         coeffs[15]=1;
```

```
            coeffs[16]=-3265;
            coeffs[17]=0;
            coeffs[18]=2741;
            coeffs[19]=0;
            coeffs[20]=-2120;
            coeffs[21]=0;
            coeffs[22]=1482;
            coeffs[23]=0;
end

generate
for(i=0; i<N; i=i+1)
begin : m
           multiplier mult(.dataa(coeffs[i]), .datab(hold[i]), .out(Add[i]));
end
endgenerate

always @(posedge clk or posedge reset)
begin
    if(reset)
        begin
                                        hold[23]            <= 0;
                                        hold[22]            <= 0;
                                        hold[21]            <= 0;
                                        hold[20]            <= 0;
                                        hold[19]            <=      0;
                                        hold[18]            <=      0;
                                        hold[17]            <=      0;
                                        hold[16]            <=      0;
                                        hold[15]    <= 0;
                                        hold[14]    <= 0;
                                        hold[13]    <= 0;
            hold[12]    <= 0;
            hold[11]    <= 0;
            hold[10]    <= 0;
            hold[9]     <= 0;
            hold[8]     <= 0;
            hold[7]     <= 0;
            hold[6]     <= 0;
            hold[5]     <= 0;
            hold[4]     <= 0;
            hold[3]     <= 0;
            hold[2]     <= 0;
            hold[1]     <= 0;
            hold[0]     <= 0;
            output_signal       <= 0;
        end
    else
        begin
```

```
                                        hold[23]                  <= hold[22];
                                        hold[22]                  <= hold[21];
                                        hold[21]                  <= hold[20];
                                        hold[20]                  <= hold[19];
                                        hold[19]                  <=      hold[18];
                                        hold[18]                  <=      hold[17];
                                        hold[17]                  <=      hold[16];
                                        hold[16]                  <=      hold[15];
                                hold[15]    <= hold[14];
                                        hold[14]    <= hold[13];
                                        hold[13]    <= hold[12];
          hold[12]    <= hold[11];
          hold[11]    <= hold[10];
          hold[10]    <= hold[9];
          hold[9]     <= hold[8];
          hold[8]     <= hold[7];
          hold[7]     <= hold[6];
          hold[6]     <= hold[5];
          hold[5]     <= hold[4];
          hold[4]     <= hold[3];
          hold[3]     <= hold[2];
          hold[2]     <= hold[1];
          hold[1]     <= hold[0];
          hold[0]     <= input_signal;
          output_signal <= (Add[0] + Add[1] +
                         Add[2] + Add[3] + Add[4] + Add[5] +
                         Add[6] + Add[7] + Add[8] + Add[9] +
                         Add[10] + Add[11] + Add[12]+ Add[13]+
                              Add[14]+ Add[15]+ Add[16]+ Add[17]
                    + Add[18]+ Add[19]+ Add[20]+ Add[21]
                    + Add[22]+ Add[23]);
        end
end
endmodule
*******#####Multiplier####********************
module multiplier (input [15:0]dataa,input [15:0]datab, output[15:0]out);
wire [31:0] result;

multi mui (.dataa(dataa),.datab(datab), .result(result));
assign out = {result[31],result[29:15]};
endmodule
```

4. Echo Machine Verilog HDL Code

```
module echo (
                                        input clk,
                                        input reset,
                                        input [15:0] input_signal,
                                        output reg [15:0] output_signal);
wire [15:0]b;
wire [15:0]c;


//assign output_signal = change_signal + c;
shiftregister SR (.clock(clk), .shiftin(output_signal), .shiftout(b));
div divider (.denom(16'b0010), .numer(b), .quotient(c));
always @(posedge clk or posedge reset)
begin
	if(reset)
		output_signal <= 0;
	else
		output_signal <= input_signal + c;


end
endmodule
```