3TB4: Lab 3

Kavya Sundaresan 400307169

Noor Al-Rajab 400291137

Group 3

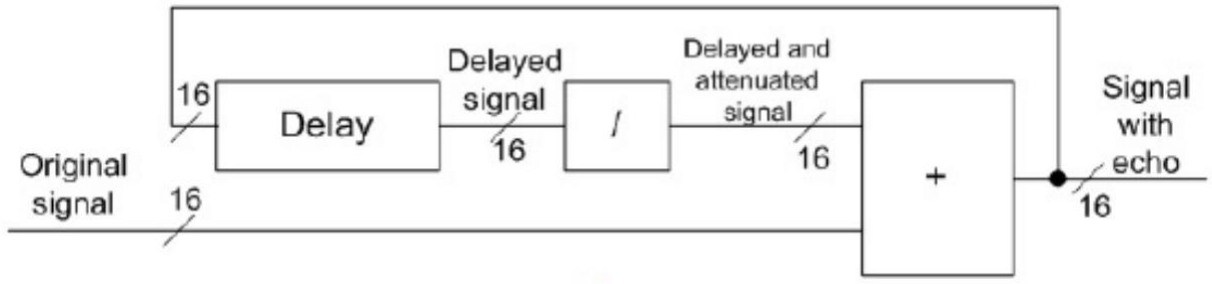
Shift Register

The shift register shifts samples by N samples, where N is the product of number of taps with tap distance. When shifting a series of sound values, a delay is produced, which will be used in the echo machine. The number of samples doesn't mean much, but since the sampling rate is known, the shift (and therefore the echo delay) in seconds can be calculated. The desired delay was something between 0.25 sec and 0.75 sec. Playing with the numbers, the number of taps and tap distance variables were eventually set to 64 and 60 respectively. The shift in samples is therefore 60*64=3840, and to get the delay value in seconds, knowing that the sampling rate is 8000 samples per second, delay=3840/8000=0.48 sec (about half a second).

```
 timescale 1 ps / 1 ps
// synopsys translate_on
module shift_register (
    clock,
    shiftin,
    shiftout,
    taps);

    input    clock;
    input [15:0]  shiftin;
    output   [15:0]  shiftout;
    output   [511:0]  taps;

    wire [15:0] sub_wire0;
    wire [511:0] sub_wire1;
    wire [15:0] shiftout = sub_wire0[15:0];
    wire [511:0] taps = sub_wire1[511:0];

    altshift_taps  ALTSHIFT_TAPS_component (
              .clock (clock),
              .shiftin (shiftin),
              .shiftout (sub_wire0),
              .taps (sub_wire1)
              // synopsys translate_off
              ,
              .aclr (),
              .clken (),
              .sclr ()
              // synopsys translate_on
              );
    defparam
        ALTSHIFT_TAPS_component.intended_device_family = "Cyclone V",
        ALTSHIFT_TAPS_component.lpm_hint = "RAM_BLOCK_TYPE=MLAB",
        ALTSHIFT_TAPS_component.lpm_type = "altshift_taps",
        ALTSHIFT_TAPS_component.number_of_taps = 64,
        ALTSHIFT_TAPS_component.tap_distance = 60,
        ALTSHIFT_TAPS_component.width = 16;

endmodule
```

Echo Machine

The way the echo machine produces the echo effect is by taking the original signal, delaying it (using the shift register) and attenuating it (done by dividing it by a value through a bitwise shift to the right) and then adding it to the original signal. That way, the output would be the original signal and a quieter, delayed version. Since it's a loop, after the original signal ends, the echo would also be fed back into the delay and attenuated, resulting in a series of echoes that gradually become quieter until silence. The division done in this project was by 4, which was done by a bitwise shift to the right by 2 bits.




```
module echo_machine (input clk, input signed [15:0] in, output signed [15:0] out);
   wire signed [15:0] delay_in;
   wire signed [15:0] delay_out;
   assign delay_in = out;
   shift_register(.clock(clk), .shiftin(delay_in), .shiftout(delay_out));

   assign out = in  + (delay_out >>> 2); //need to correct division factor

endmodule
```

Multiplier

The multiplier module multiplies 2 signed arrays of 16 bit length and returns a 32 bit output.

```
module multiplier (
    dataa,
    datab,
    result);

    input [15:0]  dataa;
    input [15:0]  datab;
    output   [31:0]  result;

    wire [31:0] sub_wire0;
    wire [31:0] result = sub_wire0[31:0];

    lpm_mult lpm_mult_component (
              .dataa (dataa),
              .datab (datab),
              .result (sub_wire0),
              .aclr (1'b0),
              .clken (1'b1),
              .clock (1'b0),
              .sclr (1'b0),
              .sum (1'b0));
    defparam
        lpm_mult_component.lpm_hint = "MAXIMIZE_SPEED=5",
        lpm_mult_component.lpm_representation = "SIGNED",
        lpm_mult_component.lpm_type = "LPM_MULT",
        lpm_mult_component.lpm_widtha = 16,
        lpm_mult_component.lpm_widthb = 16,
        lpm_mult_component.lpm_widthp = 32;


endmodule
```

Filter

The FIR filter removes a noise of 2000 Hz from the input audio. 12 coefficients were determined using MATLAB for the filter. The noisy signal elements are continuously shifted right to obtain the delayed signal. In a loop, the coefficients are multiplied with each sample of the delayed signal and the products are divided by 2^{16} to remove the applied factor in MATLAB. The divided values are continuously summed to produce the clean filtered signal.

```
module filter(input clk, input signed [15:0] noisy_signal, output signed [15:0] filtered_signal);
    integer taps = 65;
    wire signed [15:0] coeff [12:0];
    reg signed [15:0] delayed [15:0];
    wire signed [31:0] prod [12:0];
    reg signed [31:0] sum; // product of sums

    assign filtered_signal =  sum;

   assign coeff [ 0 ] = 6375;
   assign coeff [ 1 ] = 1;
   assign coeff [ 2 ] = -3656;
   assign coeff [ 3 ] = 3;
   assign coeff [ 4 ] = 4171;
   assign coeff [ 5 ] = 4;
   assign coeff [ 6 ] = 28404;
   assign coeff [ 7 ] = 4;
   assign coeff [ 8 ] = 4171;
   assign coeff [ 9 ] = 3;
   assign coeff [ 10 ] = -3656;
   assign coeff [ 11 ] = 1;
   assign coeff [ 12 ] = 6375;

    genvar i;
    generate
        for(i=0;i<=12;i=i+1) begin: multiply
            multiplier multiply(delayed[i],coeff[i],prod[i]);
        end
    endgenerate

    integer j;
    integer k;
    always @(posedge clk)
    begin
        delayed[0] <= noisy_signal;
        for(j=0;j<11;j=j+1)begin
            delayed[j+1]<=delayed[j];
        end
        for (k=0; k<=12; k=k+1) begin
            sum = sum+(prod[k]>>>15); //divides by 15 to account for MATLAB factor (2^16)
        end
    end

endmodule
```

Multiplexer

The multiplexer takes 3 inputs and uses 2 selector inputs to select one output. It first uses one selector bit to select between 2 of the inputs and passes that to a temporary variable. Then it uses the second selector bit to select between the temporary variable and the third input to obtain the final output.

```
module mux(input[15:0] a, b, c, input s1, s2, output reg[15:0] y);
   reg[15:0] temp;
   reg[15:0] dummy;
   always @(s1 or s2)
   begin
      temp[15:0] = ~s1 ? a[15:0]:c[15:0];
      y[15:0] <= ~s2 ? temp[15:0]:b[15:0];
   end
endmodule

/*
s1=0 s2=0 a
s1=0 s2=1 b
s1=1 s2=0 c
*/
```

DSP Subsystem

The purpose of this module is to have three modes (raw, filtered, echo) and to be able to change the mode using two inputs, which in this lab are switches 0 and 1. This is where the 3-to-1 multiplexer is incorporated. The multiplexers inputs were the raw signal, the output of the FIR filter, and the output of the echo machine.

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

reg signed[15:0] fir_output;

assign fir_output_w[15:0]=fir_output[15:0];

filter fir(sample_clock, input_sample[15:0], fir_output_w[15:0]); //must change to 15 instead of 12 later
echo_machine echo(sample_clock, input_sample[15:0], echo_output[15:0]);
mux mux(input_sample[15:0], fir_output[15:0], echo_output[15:0], selector[1], selector[0], output_sample[15:0]);
//assign output_sample = input_sample;



endmodule
```

1. The echo delay was 3840 samples, which is 0.48 seconds, and the division factor used was 4.
2. The filter can be redesigned to use the minimum number of coefficients which will reduce the number of multipliers needed.
3.

Flow Status	Successful - Thu Mar 07 15:02:25 2024
Quartus Prime Version	17.1.0 Build 590 10/25/2017 SJ Lite Edition
Revision Name	lab3
Top-level Entity Name	lab3
Family	Cyclone V
Device	5CSEMA5F31C6
Timing Models	Final
Logic utilization (in ALMs)	1,518 / 32,070 (5 %)
Total registers	157
Total pins	13 / 457 (3 %)
Total virtual pins	0
Total block memory bits	0 / 4,065,280 (0 %)
Total DSP Blocks	9 / 87 (10 %)
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	0 / 6 (0 %)
Total DLLs	0 / 4 (0 %)

4. A logic element is the smallest unit of logic. It consists of a 4-input LUT, a programmable register and a carry chain.
5. The multiplexer allows the user to select between the 3 audio sources (Echo, Unfiltered signal, Filtered signal) using switches 0 and 1, all of which are passed to the DSP subsystem.
6. Since the delay is 3840 samples, and each sample consists of 16 bits, the number of memory bits used is 3840*16=61,440 bits.

7. No, since the number of memory bits used in the design (61,440) was double the number of logic blocks available on the board (32,070). Although it might be possible to do a shorter delay, this greatly limits the customizability and making the delay that short might make it undetectable.