به نام او

دانشگاه صنعتی شریف

دانشکده مهندسی برق

پروژه درس ASIC/FPGA

استاد: دکتر حاج صادقی

سارا محمدی ۹۵۱۰۲۲۵۴
زینب شریفی ۹۵۱۰۱۷۵۲

تاریخ تحویل: ۱۱ تیر ۹۸

نیمسال دوم ۹۸-۱۳۹۷

I. *Voice Recorder:*

➢ Step 1: Recorder without filter

- Wecontrol: به منظور تعیین مقدار write enable در ماژول mybram این متغیر در زمان playback و زمانی که به بیت هشتم در مود record نرسیده ایم، صفر قرار داده میشود تا نوشتنی روی مموری صورت نگیرد.
- Flag: برای تشخیص اتمام record و بازگشت به خانه صفر در مموری به هنگام record مجدد یک flag نیاز است تا به کمک آن در بازگشت دوباره به record آدرس خانه حافظه صفر قرار داده شود.
- recordCounter/playbackCounter: در مود record از هر ۸ دیتای دریافت شده از ac97 یکی داخل مموری ذخیره میشود. در مود playback هم از هر آدرس مموری ۸ بار خوانده میشود و به ac97 داده میشود.
- addressRecord/addressPlayback: پوینتر هایی هستند که به خانه ای در مموری به هنگام خواندن یا نوشتن اشاره میکنند. پوینتر هر مود در مود دیگری صفر میشود.
- endAddress: در هربار record کردن آدرس خانه آخری از مموری که دیتا در آن قرار میگیرد در این متغیر ذخیره میشود تا در مود playback از خانه صفر تا این خانه خوانده شود و دوباره به اول مموری برای خواندن مجدد باز گردیم.
- Playback Mode: با شروع برنامه ابتدا در این مود قرار میگیریم در صورتی که addressRecord صفر باشد (هنوز چیزی ضبط نشده) در خروجی سیگنال tone قرار میگیرد. در غیر اینصورت خروجی ضبط شده از مموری خوانده شده و از هر آدرس ۸ بار به ac97 فرستاده میشود.(صدای ضبط شده مرتبا تکرار میشود.)
- Record Mode: در این مود از هر ۸ دیتای دریافتی یکی در مموری نوشته میشود.

در این مرحله فیلتری وجود ندارد. نتیجتا در مود playback آنچه در حافظه ذخیره شده شنیده میشود.

➢ Step 3: Recorder with filter

- فیلتر پایین گذر اعمال شده به طول ۳۱ در دو مود record و playback به ازای هر ۱ شدن ready یک ورودی جدید میگیرد و در بافر sample خود ذخیره میکند. در اینجا از circular buffer استفاده شده که با offset روی آن حرکت میکنیم. برای جمع زدن ۳۲ نمونه از وردی پس از اعمال فیلتر از تغییر index از ۰تا ۳۰ استفاده شده است. این تغییر بین ۱ شدن ۲ ready انجام و ضرب نمونه ها در ضرایب فیلتر محاسبه و جمع زده میشوند.
- در حالتی که فیلتر اعمال شود ورودی دریافتی از ac97 ابتدا به فیلتر داده میشود سپس از این ورودی فیلتر شده از هر ۸ نمونه یکی در مموری ذخیره میشود. در مود playback نیز ابتدا به ازای هر خانه حافظه ۷ تا صفر بین داده ها قرار میدهیم تا طولی مشابه قبل داشته باشیم سپس این سیگنال zero-extend شده را به فیلتر میدهیم و خروجی فیلتر به ورودی ac97 برای بخش میرود.

در نهایت کد این دو بخش به صورت زیر است:

```
module recorder(
  input wire clock,              // 100mhz system clock
  input wire reset,              // 1 to reset to initial state
  input wire playback,           // 1 for playback, 0 for record
  input wire ready,              // 1 when AC97 data is available
  input wire filter,             // 1 when using low-pass filter
  input wire [7:0] from_ac97_data, // 8-bit PCM data from mic
  output reg [7:0] to_ac97_data    // 8-bit PCM data to headphone
);
   // test: playback 750hz tone, or loopback using incoming data
   wire [19:0] tone;
   tone750hz xxx(.clock(clock),.ready(ready),.pcm_data(tone));

   wire we;
   reg wecontrol=1'b0;
   reg [2:0] recordCounter;
   reg [15:0] addressRecord=16'b0;
   reg [15:0] addressPlayback;
   wire [7:0] mem_in;
   wire [7:0] mem_out;
   reg flag=1'b0;
   reg flagPlayBack;
   wire [15:0] a;
   reg [15:0] endAddress=16'b0;
   reg [2:0] playbackCounter;
   wire FIRreset;
   //reg [4:0] FIRcounter=5'b0;
   wire filterReady;
   wire [7:0] FIRinput;
   wire [17:0] FIRoutput;

  fir31 fil(clock,FIRreset,ready,FIRinput,FIRoutput);

  mybram #(.LOGSIZE(16),.WIDTH(8))example(.addr(a),.clk(clock),.we(we),.din(mem_in),.dout(mem_out)
  assign a[15:0]=(~playback)?addressRecord:addressPlayback;

  always @(posedge clock)begin
     if(ready)begin
        if(playback)begin
           wecontrol<=1'b0;
           flag<=1'b0;
           recordCounter<=3'b0;
           to_ac97_data<=(addressRecord!=16'b0)?(filter?FIRoutput[14:7]:mem_out):tone[19:12];
           playbackCounter<=playbackCounter+3'b1;
           if(addressPlayback<addressRecord)begin
              addressPlayback<=(playbackCounter==3'b111)?(addressPlayback+16'b1):addressPlayback;
           end else begin
              addressPlayback<=16'b0;
           end
```

```
        end else begin
           playbackCounter<=3'b0;
           addressPlayback<=16'b0;
           addressRecord<=(flag)?addressRecord:16'b0;
           to_ac97_data<=mem_in;
           //--------
           //to_ac97_data<=filter ? filterOutputReady ? FIRoutput[17:10]:0: from_ac97_data;
           ..
           flag<=1'b1;
           recordCounter<=recordCounter+3'b1;
           if(recordCounter==3'b111 && addressRecord<16'hFFFF)begin
              wecontrol<=1'b1;
              addressRecord<=addressRecord+16'b1;
           end else begin
              wecontrol<=1'b0;
           end //else

        end //else

      end

    end

   assign we=(playback)?1'b0:(ready & wecontrol);
   assign mem_in=filter?FIRoutput[17:10]:from_ac97_data;
   assign FIRinput=playback?(playbackCounter!=7?0:mem_out):from_ac97_data;
   assign FIRreset=reset || !filter;
endmodule
```

```
module fir31(
  input wire clock,reset,ready,
  input wire signed [7:0] x,
  output reg signed [17:0] y
);

  // for now just pass data through
  coeffs31 coef(index,coeff);
  reg [4:0] offset=5'b11111;
  wire signed [9:0] coeff;
  wire signed [17:0] multiple;
  reg signed [7:0] sample [31:0];
  reg [4:0] index=5'b0;
  always @(posedge clock) begin
   if(reset)begin
      y<=18'b0;
      offset<=5'b11111;
      index<=5'b0;
   end else if(ready) begin
      offset<=(offset==5'b0)?5'b11111:offset-5'b1;
      sample[offset]<=x;
      y<=18'b0;
      index<=5'b0;
   end else if(index<5'b11111)begin
      index<=index+5'b1;
      y<=y+multiple;
   end

  end
  assign multiple=coeff*sample[offset-index];
endmodule
```

در نهایت مشاهده میشود با اعمال فیلتر پایین گذر صدا نرم تر شده و نویز کمتری در صدای خروجی مشاهده میشود.

- **Step 2: Test low-pass filter**

در این بخش خروجی مورد انتظار با خروجی گرفته شده از فیلتر طراحی شده مقایسه شدند.(فایل دو خروجی به همراه کد ها تحویل داده شده.) نتایج برابری میکنند.

نمونه مورد انتظار :

```
47269
42567
37809
32526
26210
18568
 9690
   69
-9553
-18432
-26077
-32398
-37691
-42464
-47184
-52040
-56828
-61006
-63891
-64924
-63891
-61006
-56828
-52040
-47184
-42464
-37691
-32398
-26077
-18432
 -9553
    69
 9690
18568
26210
32526
37808
```

نمونه گرفته شده از فیلتر طراحی شده:

```
 47269
 42567
 37809
 32526
 26210
 18568
  9690
    69
 -9553
-18432
-26077
-32398
-37691
-42464
-47184
-52040
-56828
-61006
-63891
-64924
-63891
-61006
-56828
-52040
-47184
-42464
-37691
-32398
-26077
-18432
 -9553
    69
 9690
18568
26210
32526
```

II. *Pitch Shifter:*

- ایده اصلی استفاده شده در این بخش استفاده از دو بافر به طور همزمان در هر مود switch هاست به گونه ای که در یکی از آن ها از ac97 بخوانیم و تغییرات لازم را انجام دهیم درحالی که از رجیستر دیگر به ac97 ورودی میدهیم.
- به این منظور ۳ مدل counter تعریف شده که بسته به مود switch ها از آن ها استفاده بهینه میشود. counter۲۵۶ تایی همواره برای دادن داده به ac97 استفاده میشود چون فرض گرفته شده که داده ها در بازه های ۲۵۶تایی process شوند. Counter ۱۲۸ تایی برای نصف کردن فرکانس و counter ۶۴ تایی برای به ۴ تقسیم کردن فرکانس استفاده شده.
- در هر مود وقتی برای بار اول وارد میشویم دیتایی برای فرستادن به ac97 در همان لحظه وجود ندارد بنابراین فقط از ورودی دیتا گرفته شده و تغییرات را اعمال میکنیم. بعد از ۲۵۶ تای اول flag مربوط به این بخش(first-time) ۱ شده و وارد حالت همزمانی خواندن و نوشتن میشویم.
- برای جابجایی بین مود های مختلف از یک case استفاده شده که هر case در ابتدا flag بخش های دیگر را صفر میکند.
- در مود هایی که قرار است فرکانس را افزایش دهیم با توجه به عدد داده شده دیتا ۲۵۶ تایی را عینا تکرار میکنیم سپس یکی در میان(برای ۲) یا ۴تا یکی(برای ۴) نمونه برداری میکنیم و به خروجی میدهیم.
- در مودهایی که قرار است فرکانس را کاهش دهیم با توجه به عدد داده شده نصف یا یک چهارم دیتا را نگه داشته و سپس در خروجی در بین آن ها صفر قرار میدهیم(یکی یا سه تا)
- در ماژول recorder ماژولpitchshifter صدا زده میشود. در این ماژول یک reset هم برای pitchshifter تعریف شده. با هربار وارد شدن به مود record ماژول pitchshifter یک بار reset میشود و همه counter ها برابر صفر قرار میگیرند.
- مشاهده شد که با افزایش فرکانس صدا زیر و با کاهش آن صدا بم میشود. البته برای ۴ برابر یا یک چهارم کردن فرکانس نویز بیشتر میشود.

کد این بخش به صورت زیر است :

```
module pitchShifter(
    input wire ready,
    input wire[7:0] x,
    input wire reset,
    input wire [2:0] switch,
    input wire clk,
    output reg [7:0] y
    );

     //----------
     //wire [2:0] switch;
     //assign switch=3'b010;
     //-------
     reg [7:0] sample1 [1023:0];
     reg [7:0] sample2 [1023:0];
     reg [7:0] sample1_d [255:0];
     reg [7:0] sample2_d [255:0];
     reg whichSample=1'b0;
     reg firstTime1=1'b0;
     reg firstTime2=1'b0;
     reg firstTime3=1'b0;
     reg firstTime4=1'b0;
     reg [7:0] counter256_1=8'b0;
     reg [7:0] counter256_2=8'b0;
     reg [6:0] counter128_1=7'b0;
     reg [6:0] counter128_2=7'b0;
     reg [5:0] counter64_1=6'b0;
     reg [5:0] counter64_2=6'b0;
     reg [2:0] changeMode;
     always @(posedge clk)begin
      if(reset)begin
                firstTime1<=1'b0;
                firstTime2<=1'b0;
                firstTime3<=1'b0;
                firstTime4<=1'b0;
                counter256_1<=8'b0;
                counter256_2<=8'b0;
                counter128_1<=7'b0;
                counter128_2<=7'b0;
                counter64_1<=6'b0;
                counter64_2<=6'b0;
                changeMode<=switch;
      end else if(ready)begin
        case(switch)
            3'b000:begin
                y<=x;
                firstTime1<=1'b0;
                firstTime2<=1'b0;
                firstTime3<=1'b0;
                firstTime4<=1'b0;
                counter256_1<=8'b0;
                counter256_2<=8'b0;
                counter128_1<=7'b0;
                counter128_2<=7'b0;
                counter64_1<=6'b0;
                counter64_2<=6'b0;
            end
            3'b001:begin //2-increase
                firstTime3<=1'b0;
                firstTime2<=1'b0;
                firstTime4<=1'b0;
                if(changeMode!=3'b001)begin
                    counter256_1<=8'b0;
                    counter256_2<=8'b0;
                    counter128_1<=7'b0;
                    counter128_2<=7'b0;
                    counter64_1<=6'b0;
                    counter64_2<=6'b0;
                    changeMode<=3'b001;
                end else if(firstTime1==1'b0)begin
                    sample1[counter256_1]<=x;
                    sample1[counter256_1+9'd256]<=x;
                    counter256_1<=counter256_1+8'b1;
                    firstTime1<=(counter256_1==8'd255)?1'b1:1'b0;
                end else if(~whichSample) begin
                    counter256_1<=8'b0;
                    y<=sample1[2'd2*counter256_2];
                    counter256_2<=counter256_2+8'b1;
                    sample2[counter256_2]<=x;
                    sample2[counter256_2+9'd256]<=x;
                    whichSample<=(counter256_2==8'd255)?1'b1:1'b0;
                end else begin
                    counter256_2<=8'b0;
                    y<=sample2[2'd2*counter256_1];
                    counter256_1<=counter256_1+8'b1;
                    sample1[counter256_1]<=x;
                    sample1[counter256_1+9'd256]<=x;
                    whichSample<=(counter256_1==8'd255)?1'b0:1'b1;
                end

            end //end 3'b001
            3'b010:begin  //4 increase
                firstTime3<=1'b0;
                firstTime1<=1'b0;
                firstTime4<=1'b0;
                if(changeMode!=3'b010)begin
                    counter256_1<=8'b0;
                    counter256_2<=8'b0;
                    counter128_1<=7'b0;
                    counter128_2<=7'b0;
                    counter64_1<=6'b0;
                    counter64_2<=6'b0;
                    changeMode<=3'b010;
                end else if(firstTime2==1'b0)begin
                    sample1[counter256_1]<=x;
                    sample1[counter256_1+9'd256]<=x;
                    sample1[counter256_1+10'd512]<=x;
                    sample1[counter256_1+10'd768]<=x;
                    counter256_1<=counter256_1+8'b1;
                    firstTime2<=(counter256_1==8'd255)?1'b1:1'b0;
                end else if(~whichSample) begin
                    counter256_1<=8'b0;
                    y<=sample1[3'd4*counter256_2];
                    counter256_2<=counter256_2+8'b1;
                    sample2[counter256_2]<=x;
                    sample2[counter256_2+9'd256]<=x;
                    sample2[counter256_2+10'd512]<=x;
                    sample2[counter256_2+10'd768]<=x;
                    whichSample<=(counter256_2==8'd255)?1'b1:1'b0;
                end else begin
                    counter256_2<=8'b0;
                    y<=sample2[3'd4*counter256_1];
                    counter256_1<=counter256_1+8'b1;
                    sample1[counter256_1]<=x;
                    sample1[counter256_1+9'd256]<=x;
                    sample1[counter256_1+10'd512]<=x;
                    sample1[counter256_1+10'd768]<=x;
                    whichSample<=(counter256_1==8'd255)?1'b0:1'b1;
                end

            end
            3'b100:begin  //2 decrease
                firstTime1<=1'b0;
                firstTime2<=1'b0;
                firstTime4<=1'b0;
                if(changeMode!=3'b100)begin
                    counter256_1<=8'b0;
                    counter256_2<=8'b0;
                    counter128_1<=7'b0;
                    counter128_2<=7'b0;
                    counter64_1<=6'b0;
                    counter64_2<=6'b0;
                    changeMode<=3'b100;
                end else if(firstTime3==1'b0)begin
                    sample1_d[2*counter128_1]<=x;
                    sample1_d[2*counter128_1+7'b1]<=8'b0;
                    counter128_1<=counter128_1+7'b1;
                    firstTime3<=(counter128_1==8'd127)?1'b1:1'b0;
                end else if(~whichSample) begin
                    counter128_1<=7'b0;
                    counter256_2<=8'b0;
                    y<=sample1_d[counter256_1];
                    counter256_1<=counter256_1+8'b1;
                    if(counter128_2<=7'd127)begin
                        sample2_d[2*counter128_2]<=x;
                        sample2_d[2*counter128_2+7'b1]<=8'b0;
                    end
                    counter128_2<=counter128_2+7'b1;
                    whichSample<=(counter256_1==8'd255)?1'b1:1'b0;
                end else begin
```

```
            3'b100:begin  //2 decrease
                firstTime1<=1'b0;
                firstTime2<=1'b0;
                firstTime4<=1'b0;
                if(changeMode!=3'b100)begin
                    counter256_1<=8'b0;
                    counter256_2<=8'b0;
                    counter128_1<=7'b0;
                    counter128_2<=7'b0;
                    counter64_1<=6'b0;
                    counter64_2<=6'b0;
                    changeMode<=3'b100;
                end else if(firstTime3==1'b0)begin
                    sample1_d[2*counter128_1]<=x;
                    sample1_d[2*counter128_1+7'b1]<=8'b0;
                    counter128_1<=counter128_1+7'b1;
                    firstTime3<=(counter128_1==8'd127)?1'b1:1'b0;
                end else if(~whichSample) begin
                    counter128_1<=7'b0;
                    counter256_2<=8'b0;
                    y<=sample1_d[counter256_1];
                    counter256_1<=counter256_1+8'b1;
                    if(counter128_2<=7'd127)begin
                        sample2_d[2*counter128_2]<=x;
                        sample2_d[2*counter128_2+7'b1]<=8'b0;
                    end
                    counter128_2<=counter128_2+7'b1;
                    whichSample<=(counter256_1==8'd255)?1'b1:1'b0;
                end else begin
                    counter128_2<=7'b0;
                    counter256_1<=8'b0;
                    y<=sample2_d[counter256_2];
                    counter256_2<=counter256_2+8'b1;
                    if(counter128_1<=7'd127)begin
                        sample1_d[2*counter128_1]<=x;
                        sample1_d[2*counter128_1+7'b1]<=8'b0;
                    end
                    counter128_1<=counter128_1+7'b1;
                    whichSample<=(counter256_2==8'd255)?1'b0:1'b1;
                end
            end
            3'b101:begin  //4 decrease
                firstTime1<=1'b0;
                firstTime2<=1'b0;
                firstTime3<=1'b0;
                if(changeMode!=3'b101)begin
                    counter256_1<=8'b0;
                    counter256_2<=8'b0;
                    counter128_1<=7'b0;
                    counter128_2<=7'b0;
                    counter64_1<=6'b0;
                    counter64_2<=6'b0;
                    changeMode<=3'b101;
                end else if(firstTime4==1'b0)begin
                    sample1_d[4*counter64_1]<=x;
                    sample1_d[4*counter64_1+6'b1]<=8'b0;
                    sample1_d[4*counter64_1+6'd2]<=8'b0;
                    sample1_d[4*counter64_1+6'd3]<=8'b0;
                    counter64_1<=counter64_1+6'b1;
                    firstTime4<=(counter64_1==6'd63)?1'b1:1'b0;
                end else if(~whichSample) begin
                    counter64_1<=6'b0;
                    counter256_2<=8'b0;
                    y<=sample1_d[counter256_1];
                    counter256_1<=counter256_1+8'b1;
                    if(counter64_2<=6'd63)begin
                        sample2_d[4*counter64_2]<=x;
                        sample2_d[4*counter64_2+6'd1]<=8'b0;
                        sample2_d[4*counter64_2+6'd2]<=8'b0;
                        sample2_d[4*counter64_2+6'd3]<=8'b0;
                    end
                    counter64_2<=counter64_2+6'b1;
                    whichSample<=(counter256_1==8'd255)?1'b1:1'b0;
                end else begin
                    counter64_2<=6'b0;
                    counter256_1<=8'b0;
                    y<=sample2_d[counter256_2];
                    counter256_2<=counter256_2+8'b1;
                    if(counter64_1<=6'd63)begin
                        sample1_d[4*counter64_1]<=x;
                        sample1_d[4*counter64_1+6'd1]<=8'b0;
                        sample1_d[4*counter64_1+6'd2]<=8'b0;
                        sample1_d[4*counter64_1+6'd3]<=8'b0;
                    end
                    counter64_1<=counter64_1+6'b1;
                    whichSample<=(counter256_2==8'd255)?1'b0:1'b1;
                end
            end
            default:y<=x;



        endcase

      end
     end


endmodule
```