The Implementation of Delta-Sigma Modulation in Digital-to-Analog Converter

CPRE583 2011 Fall Project Report

Team Member:

Xin Zhao

Mengduo Ma

Chi Chen Fang

Kuan Hsing Ho

1. Project Idea

The motivation of this project is from the study of EE 505 CMOS Data Conversion circuits. In that class, we studied a type of digital-to-analog data converter which is used Delta-Sigma algorithm. That class is almost analog circuit design, but this kind of DAC can be implemented in digital, except for an analog low pass filter.

2. The Overview of the Project

Our project aims at the implementation of delta-sigma modulation in digital to analog converter. We have investigated a series of techniques to improve the performance of digital audio converter in terms of power consumption and noise shaping capability.

This audio DAC has been successfully applied in digital amplifier field. We used the Matlab Simulink tool to simulate the algorithm. And the simulation results had been verified using Xilinx's FPGA.

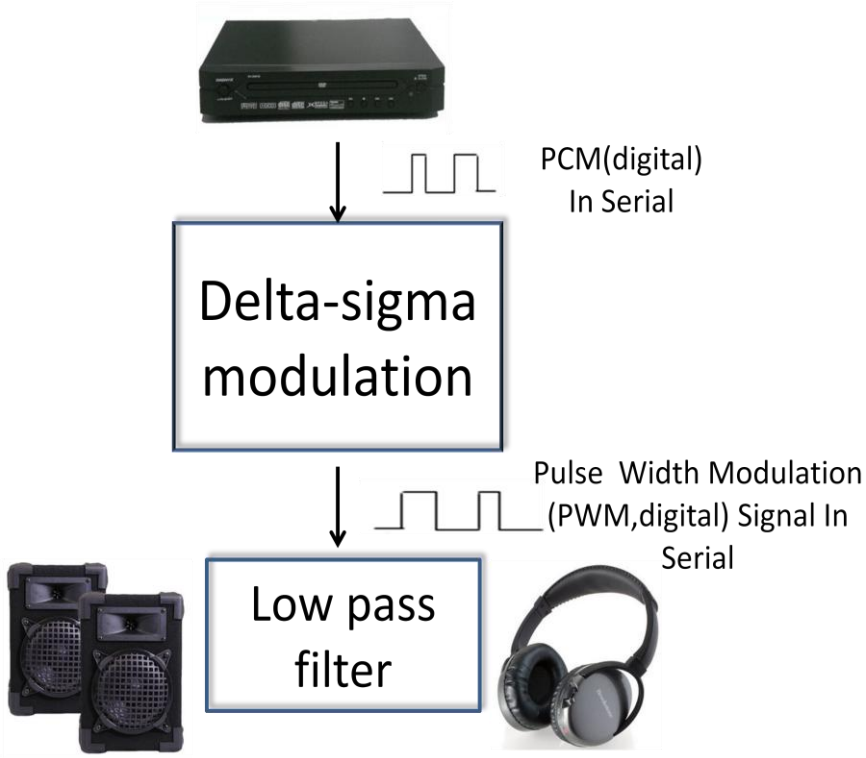
Our group mainly works on the following parts:

1) We used delta-signal modulator to push noise in audio to high frequency band, and then we adopt the speaker/headphone as the low-pass filter to filtrate high frequency white noise.
2) We reduced the dynamic power consumption of the whole audio amplifier system by decreasing the system frequency to 16.9344MHz. It can not be reduced anymore. Otherwise, the synchronization with DVD player will be a problem. We also minimize the frequency of every module to save power.
3) We designed and added a sleep mode module, which reduce the power consumption of the whole audio DAC system in case that it has not been in use for a long time (3 seconds), when considering possible use in portable digital audio field.
4) For additional functions, we added a digital volume controller in this audio amplifier system.

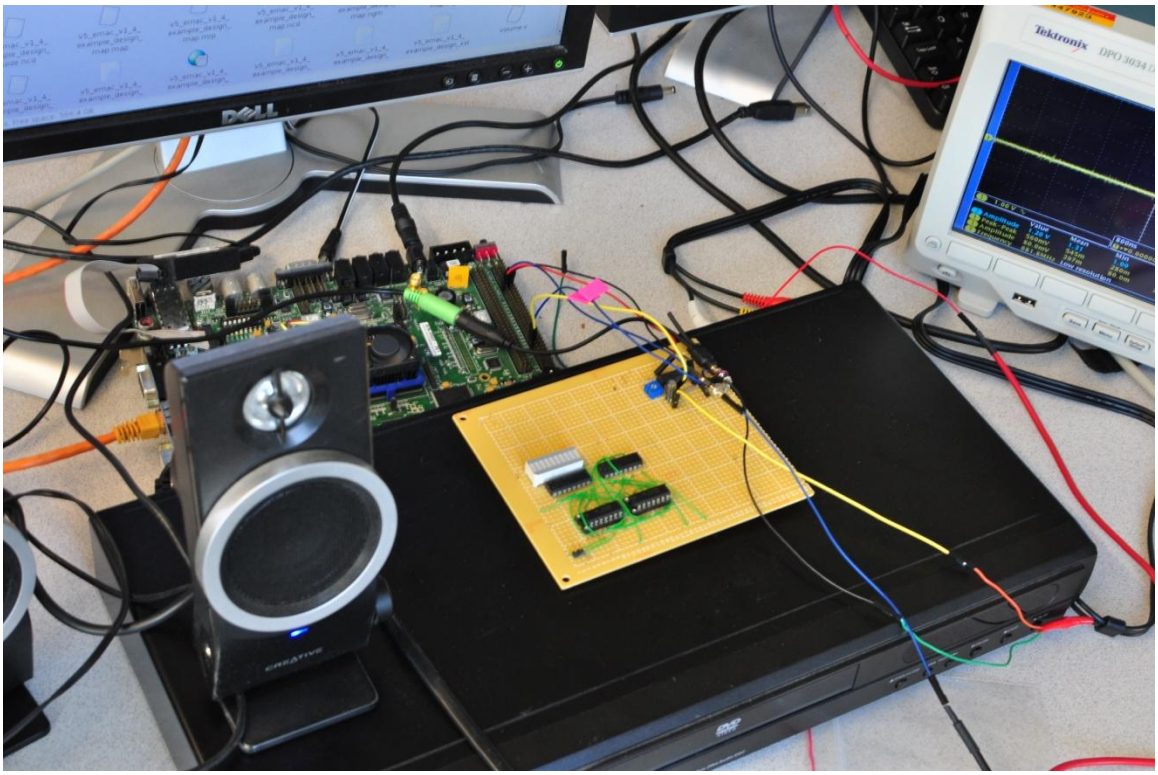
Our project decreased the power consumption and enhanced noise shaping capability of the whole audio system. As the result of the project, using two-stage noise shaping and 16 times oversampling, the SNR can reach about 100dB in theory. But we did not get a chance to measure it because we had not found equipments.

3. The Overview of the System

The design has the ability to process stereo digital audio signal form DVD player through SPDIF interface.



The real system is shown below, including DVD player, FPGA and speaker.



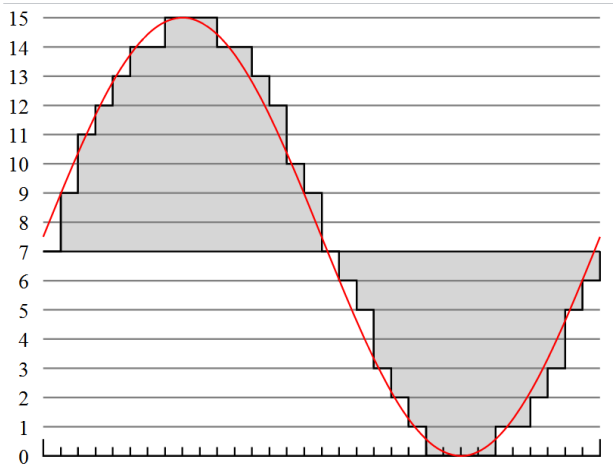
4. The Demo of the Project.

http://www.youtube.com/watch?v=582E_OXeVOU

5. High-Level Algorithm

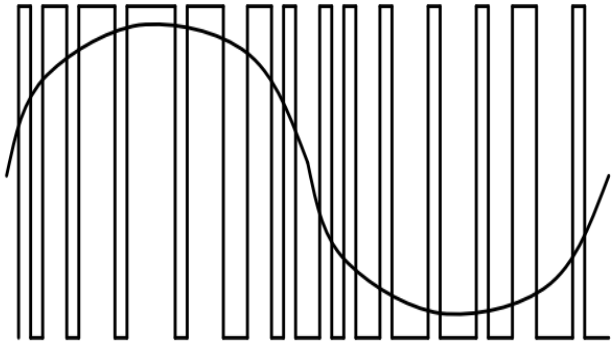
5.1 Input Signal--PCM: quantization of analog signal

The PCM signal is 16-bit wide and contains analog signal information, which is widely used in digital communication field.



5.2 Output--PWM: digital signal but also has analog information

As shown below, the width of this kind of digital signal is keep changing. So this signal charges or discharges the output low-pass filter or speaker/headphone, so that the analog signal can be restored.



5.3 Noise shaping

Noise shaping is a negative feedback of the digital signal. It plays the role of feeding back the former data to the consequent data and thus eliminates the quantization error. Figure 5-1 shows the model of Non-noise shaping. Figure 5-2 shows the model of one stage noise shaping.

In figure 5-2, the system feedbacks the output Y of quantizer. At the same time, it feeds back the input signal of the quantizer to the original input X. Which means, the system feedbacks both the input (U) and output (Y) signals of the quantizer. The difference between the Y and U is quantization error.

We deliver the sum of response part of former signal with the consequent signal to the quantizer, as well as respond it to the input port, get the quantization error of this time and feed it back to the signal.

If we view this process mathematically, this process is differential the quantization error. It subtracts quantization error of last time (Q_{n-1}) from the quantization error of current time (Q_n). Therefore, it decreases the quantization error time by times.

The distortion caused by quantization error distributes equally in the full frequency band. After the noise shaping, which works similar to high pass filter, the noise in audible band decreases sharply, while the noise out of audible band increases sharply. Such procedure greatly changes the shape of noise distribution, hence the name is noise shaping.

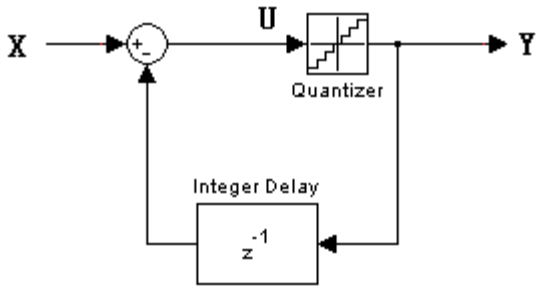


Figure 5-1 Non-noise shaping model

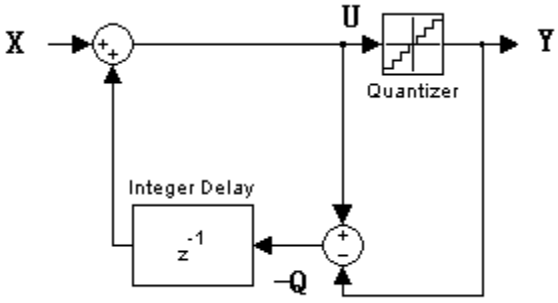


Figure 5-2 One-stage noise shaping model

The noise shaping circuits can decrease the error of the quatization. The speed of quantizer should be as fast as possible to feed a data (16 bits) back multiply. Therefore, we need to greatly increase the sampling times, even to tens or hundreds of times. For example, the speed of quantizer is 50MHz, the data speed is 44110/s, that is, the data can be fed back 50×1000000/44100 = 1134 times. Namely, we feed one data back as many as 1000 times. Now, we analyze the process using mathematical expression.

X: 16-bit input signals

Y: output signals of the system

Q: quantization error

Z^{-1} : time delay unit, signal delayed for one cycle

U : the input signal of quantizer

In figure 5-1， Y is subtracted from X repeatedly after Y is fed back. Z^{-1} delays Y for once cycle. $X-Y \cdot Z^{-1}=U$. Therefore X is different from U.

In figure 5-2, the system feeds back Y, as well as U to the input port. Because it is feedbacked before time delay unit, after subtracting Y, U is feedbacked to input port through time delay unit. U-Y is the difference between quantizer input and quantizer output. The difference is called quantization error.

$$Q=Y-U \tag{5-1}$$

$$U-Y=-Q \tag{5-2}$$

Feed back –Q to the input port, add with X to get U,

$$U=X-Q \tag{5-3}$$

If we view from the time sequence, the feedbacked U is one time cycle delayed than Q. So we need to use n to distinguish. The current time is n, so the one of last time is n-1. And we denote the Q of last time as $Q_{(n-1)}$.

$$U_{(n)}=X_{(n)}-Q_{(n-1)} \tag{5-4}$$

$$Q=Y-U \tag{5-5}$$

$$Q_{(n)}=Y_{(n)}-U_{(n)} \tag{5-6}$$

Combine with (5-4) and (5-6), we get the relationship between X and Y

$$Y_{(n)}-Q_{(n)}=X_{(n)}-Q_{(n-1)} \tag{5-7}$$

$$\begin{aligned} Y_{(n)} &= X_{(n)}-Q_{(n-1)}+Q_{(n)} \\ &= X_{(n)}+Q_{(n)}-Q_{(n-1)} \end{aligned} \tag{5-8}$$

$Q_{(n)}-Q_{(n-1)}$ is the difference of quantization error of current time and last time. So 5-8 means the input subtract the quantization error times by times.

$$Q_{(n)}-Q_{(n-1)}=(1-Z^{-1}) \cdot Q \tag{5-9}$$

That is

$$Y = X + (1 - Z^{-1}) \cdot Q \tag{5-10}$$

This is the generalized equation of noise shaping. $1 - Z^{-1}$ is the differential. This equation means that the input of one stage noise shaping is the sum of input and quantization error after difference. Therefore the noise in audible range is reduced, the noise out of audible range is increased, changes the shape of noise distribution, namely, noise shaping.

5.4 Simulation analysis of two-stage noise shaping

Two-stage noise shaping model is shown in the figure 5-3.There is still one quantizer. "Feed back before and behind quantizer" and "quantization error is 180^{o} fed back" are the same as in one-stage noise shaping. But we added a time delay unites on the front side of model, it also feeds back signal to input port. So the output of these two time delay units will be amplified by 2 times and -1 time and then fed back. R2 stands for quantization error of one delay. R1 stands for quantization error of two delays. Same as one-stage integrated model, list the expression of input port of quantizer and we can find U is equal to -1 R1 plus 2 R2 , i.e.

$$U = X - R_1 + 2R_2 \tag{5-11}$$

As R2 is equal to one delay -Q, R1 means delayed again, so

$$R_2 = Z^{-1} \cdot (-Q) \tag{5-12}$$

$$R_1 = Z^{-1} \cdot R_2 = Z^{-1} \cdot (Z^{-1} \cdot (-Q)) = (Z^{-1})^2 \cdot (-Q) \tag{5-13}$$

Thus U can be expressed as

$$\begin{aligned} U &= X - (Z^{-1})^2 \cdot (-Q) + 2 Z^{-1} \cdot (-Q) \\ &= X + (Z^{-1})^2 \cdot Q - 2 Z^{-1} \cdot Q \end{aligned} \tag{5-14}$$

Outputting Y is same as one-stage noise shaping, and also can be expressed as:

$$Y = U + Q \tag{5-15}$$

$$U = Y - Q \tag{5-16}$$

So

$$Y - Q = X + (Z^{-1})^2 \cdot Q - 2 Z^{-1} \cdot Q \tag{5-17}$$

$$\begin{aligned} Y &= Q + X + (Z^{-1})^2 \cdot Q - 2 Z^{-1} \cdot Q \\ &= X + [1 - 2 Z^{-1} + (Z^{-1})^2] \cdot Q \end{aligned}$$

$$= X + (1 - Z^{-1})^{2} \cdot Q \qquad (5\text{-}18)$$

As $1 - Z^{-1}$ means differential, its square means twice difference. So two-stage noise shaping exactly is the process of adding the quantization error after twice difference into input port.

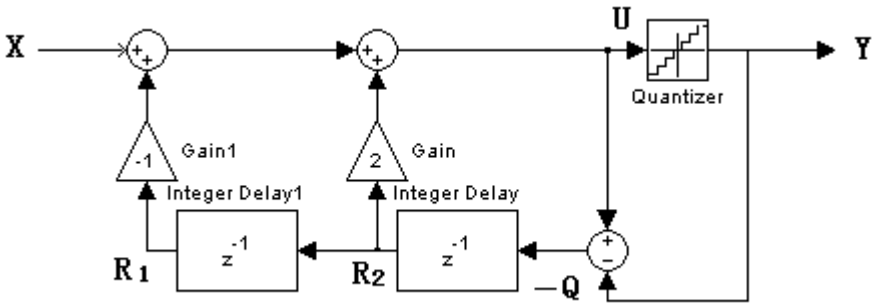


Figure 5-3 Two-stage noise shaping model

We use Simulink in Matlab software environment to make the simulation, and the simulation model is shown as in figure 5-4, the input is white noise. We observe the frequency spectrum of noise both in input port and output port(as seen in figure 5-5).We can find that curve of the original white noise with tape limit upsweeps in high band after two-stage noise shaping. Therefore the noise in audible range is reduced, the noise out of audible range is increased, changes the shape of noise distribution, namely, noise shaping.

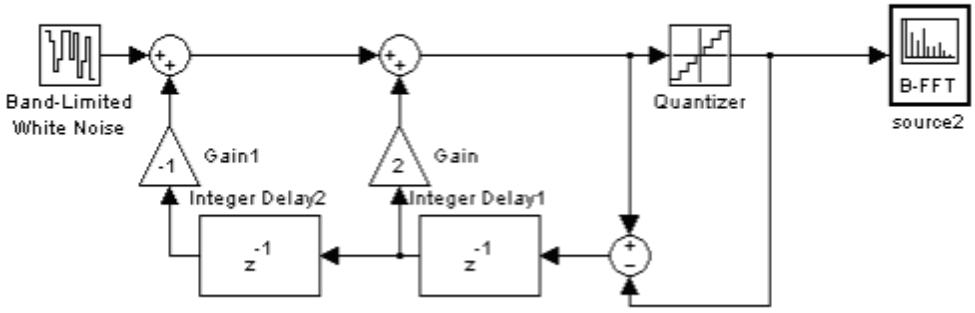
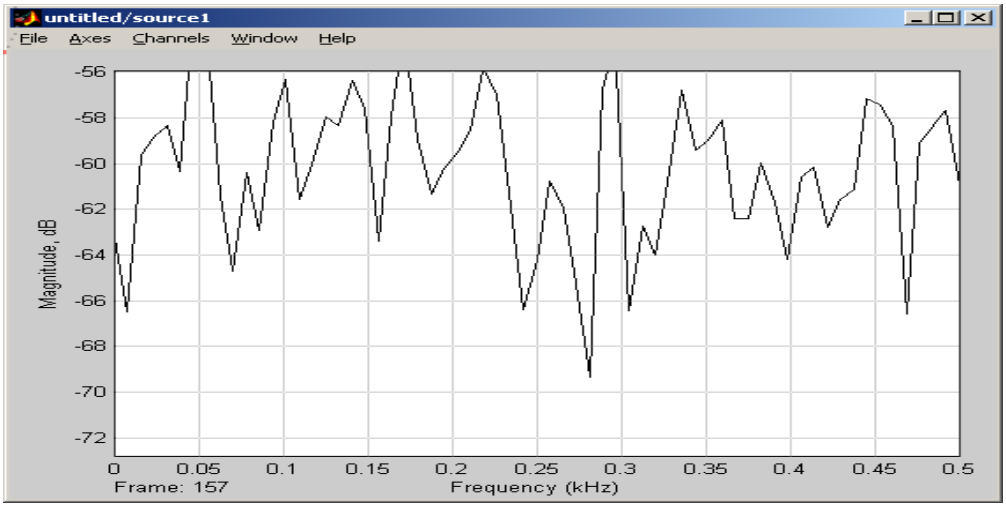
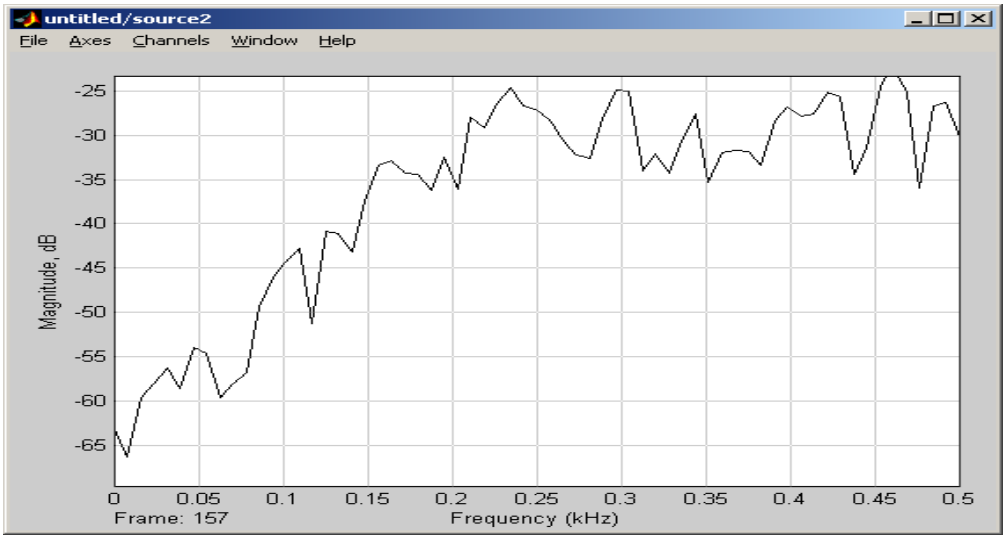


Figure 5-4 Two stage noise shaping model used for simulation



(a) Input flat noise frequency



（b）two stage noise shaping output frequency

Figure5-5 The simulation result of two stage noise shaping model

5.5 Simulation analysis of multi-stage noise shaping

The elemental mode of multi-stage noise shaping is shown as in figure5-6, we set it as 3-stage. MASH（Multi-stage noise Shaping）can be classified as first level, second level and third level. Every single level is as same as the elemental mode of one-stage noise shaping model. The first level is totally same as one-stage noise shaping model; One more time differential quantization on quantization error of output of first level forms the second level; implement differential quantization on the quantization error of second level. Therefore, the quantified output of second level and third level gets the difference processed and then adds output of first level, turns to be the final output.

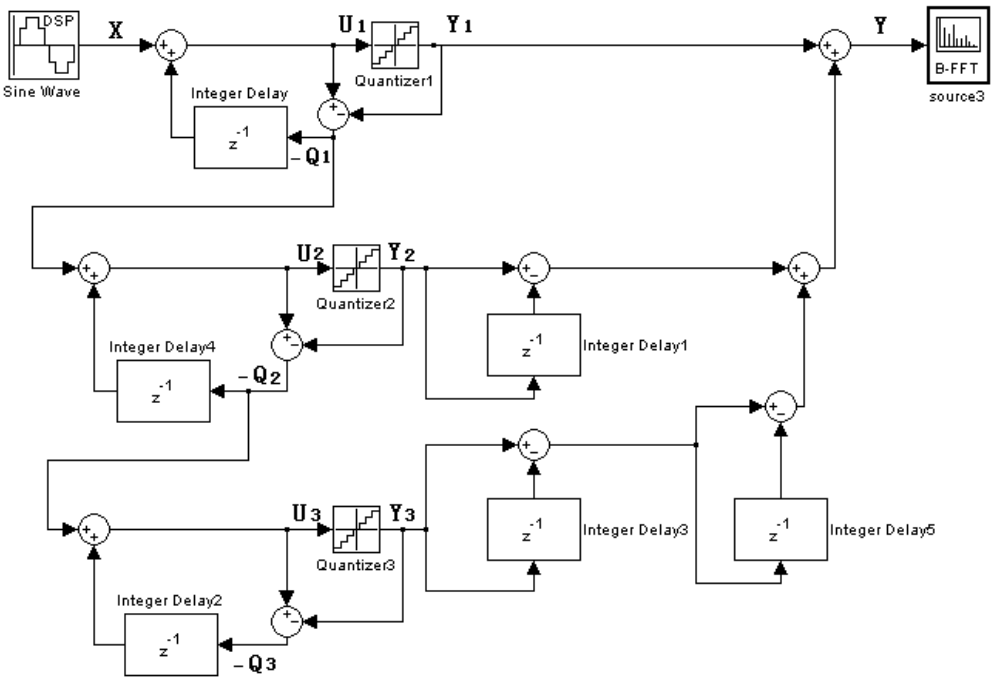


Figure 5-6 Three stages noise shaping model

As the output of first level is exactly the same as in one-stage noise shaping model, so

$$Y_1 = X + (1 - Z^{-1}) \cdot Q \qquad (5\text{-}19)$$

It's same in second level and third level, but the inputs are respectively opposite numbers of quantization error of previous level, i.e. –Q, replace the X by –Q, and therefore the Y_2、Y_3 will be

$$Y_2 = (-Q_1) + (1 - Z^{-1}) \cdot Q_2 \qquad (5\text{-}20)$$

$$Y_3 = (-Q_2) + (1 - Z^{-1}) \cdot Q_3 \qquad (5\text{-}21)$$

We can compute as top for fourth level and fifth level. The Q_2 and Q_3 in the expression are respectively the quantization error of the second level and third level.

Second level and third level have differential attached, besides, the third level have 2 differentials. As mentioned before, differential in digit circuit exactly means subtracting previous value, represent as $1 - Z^{-1}$. Thus, Y_2 in n times minus Y_2 in n-1 times and then adds Y_1 to be the second level; the output of third level get 2 times difference processed, like we mentioned in two-stage noise shaping which can be represented by $(1\text{-}Z^{-1})^2$.

$$Y = Y_1 + (1 - Z^{-1}) \cdot Y_2 + (1 - Z^{-1})^2 \cdot Y_3 \qquad (5\text{-}22)$$

substitute Y_2、Y_3 into above expression， and we can get

$$Y=X+(1-Z^{-1})\cdot Q_1+$$

$$(1-Z^{-1})\cdot[(-Q_1)+(1-Z^{-1})\cdot Q_2]+$$

$$(1-Z^{-1})^2\cdot[(-Q_2)+(1-Z^{-1})\cdot Q_3]$$

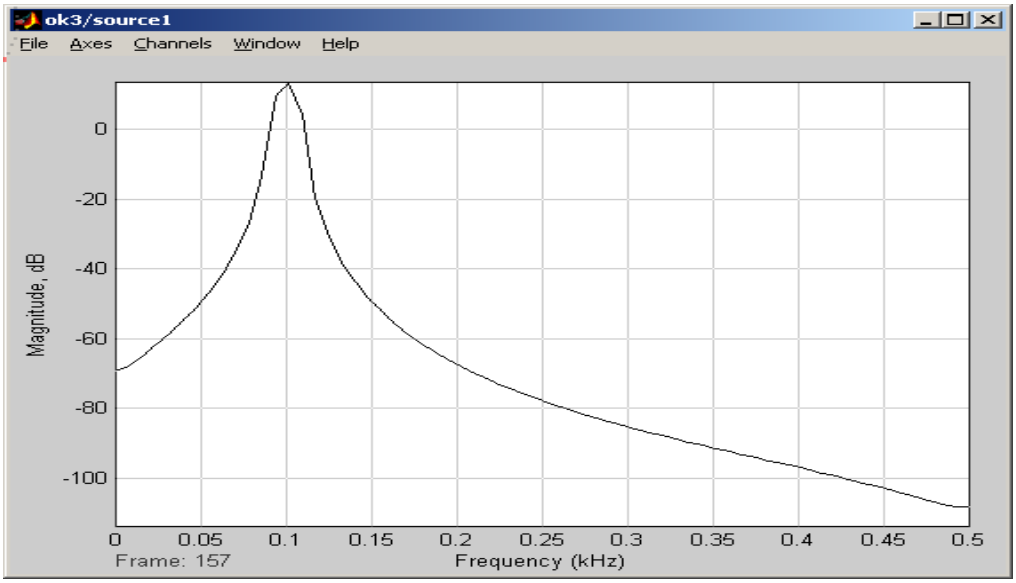
$$=X+(1-Z^{-1})\cdot Q_1-(1-Z^{-1})\cdot Q_1+(1-Z^{-1})^2\cdot Q_2-(1-Z^{-1})^2\cdot Q_2+$$

$$(1-Z^{-1})^3\cdot Q_3 \qquad (5\text{-}23)$$

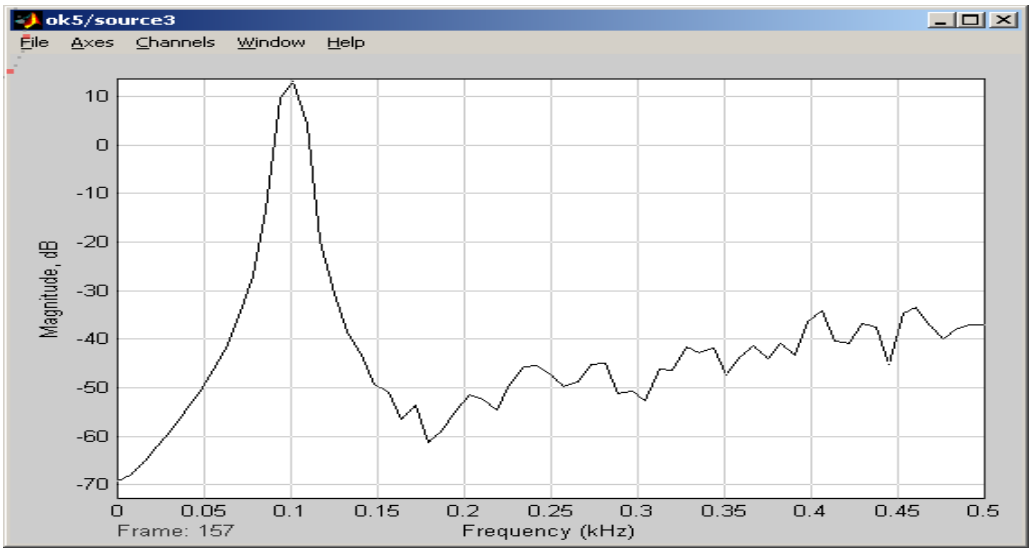
Q_1、Q_2 are canceled out， so

$$Y=X+(1-Z^{-1})^3\cdot Q_3 \qquad (5\text{-}24)$$

After being processed in differential, the second level and third level will make the Q_1 and Q_2 be canceled out. At last the expression will be this form: Q_3 was differentiated by 3 times and then add X. Three-stage noise shaping can make the distribution character of quantizing noise 18(db/double frequency) curve. Simulation result of three-stage noise shaping model is as seen in figure 5-7. We can find that quantization error is introduced due to quantification. But the distribution of noise (the quantization error), is characterized. As we can see, the curve of the original white noise should be distributed in the full frequency band. But the noises of audible region reduce rapidly, and the noise outside of audible region increases greatly. The shape of noise distribution accordingly changes a lot.



(a) Input signal spectrum of three stage noise shaping



(b) Output signal spectrum of three stage noise shaping

Figure 5-7 Simulation result of three stage noise shaping model

The figure 5-8 and figure 5-4 both are two-stage noise shaping model, the difference is their input signal. Input signal in figure 5-4 is white noise with band limit, and figure 5-8 input signal is sine-wave. Figure 5-9 is the frequency spectrum of input signal of two-stage noise shaping model. Compare this with figure 5-7, you can find that three-stage noise shaping can obtain better SNR.

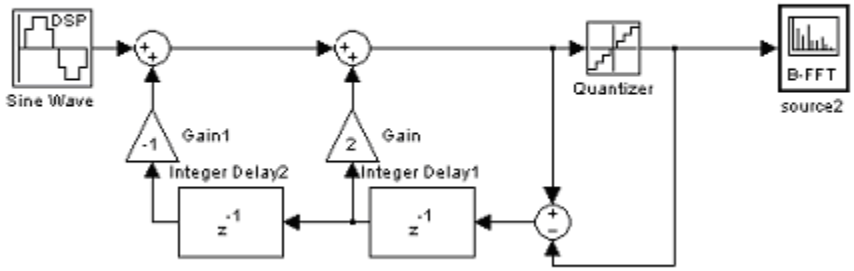


Figure 5-8 Two stage noise shaping with the stage sine input signal

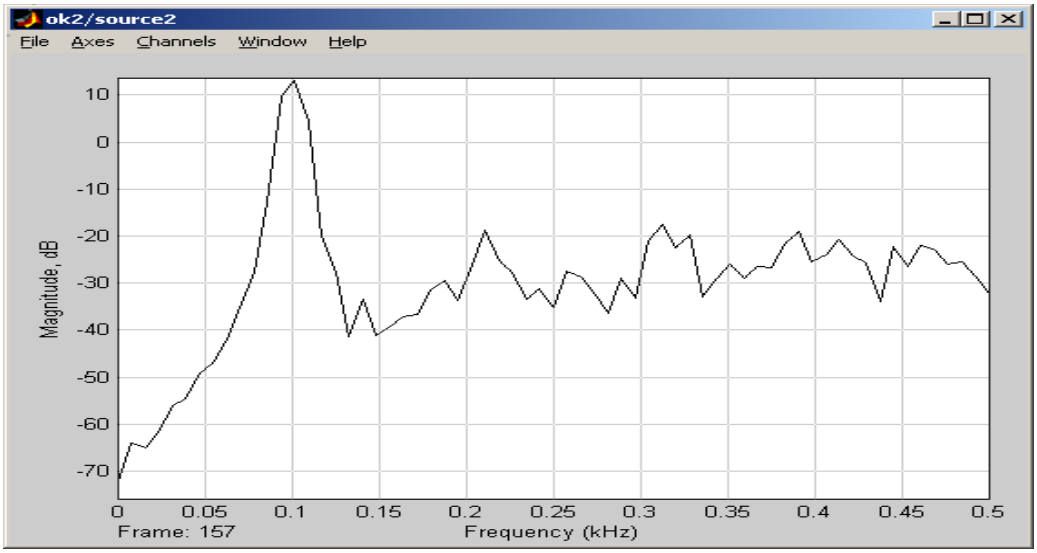


Figure 5-9 Output signal of two stage noise shaping with the stage sine input signal

The two-stage three times noise shaping is shown as follows.

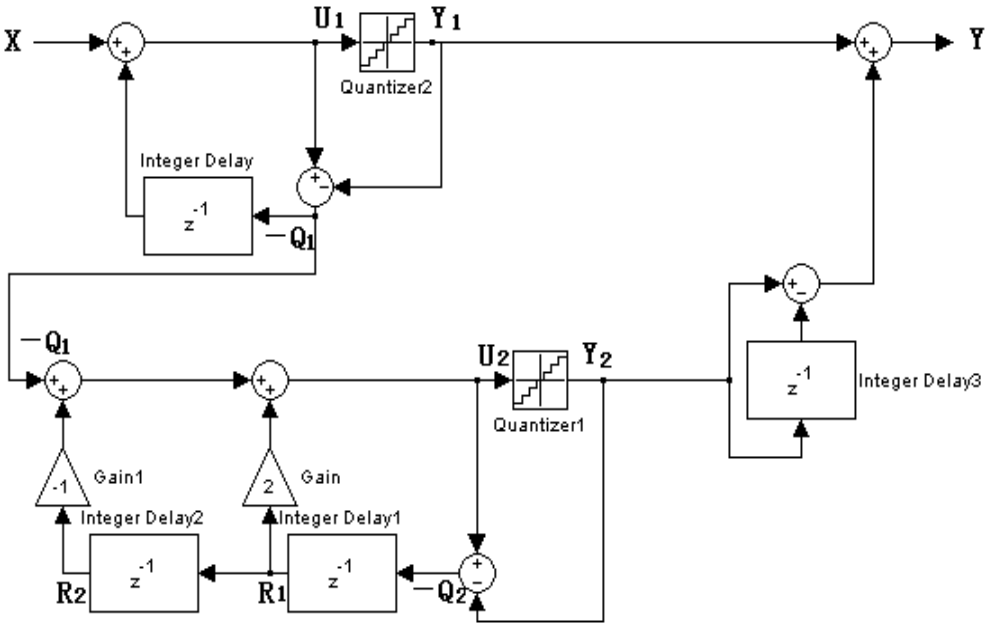


Figure 5-10 Two-stage three times noise shaping model

The first stage is one stage noise shaping

$$Y_1 = X + (1 - Z^{-1}) \cdot Q_1 \qquad (5\text{-}25)$$

The second stage is two stage noise shaping

$$Y_2 = -Q_1 + (1-Z^{-1})^2 \cdot Q_2 \qquad (5\text{-}26)$$

After the difference,

$$(1-Z^{-1}) \cdot Y_2 = -(1-Z^{-1}) \cdot Q_1 + (1-Z^{-1})^3 \cdot Q_2 \qquad (5\text{-}27)$$

Therefore, the output Y is

$$\begin{aligned} Y &= Y_1 + (1-Z^{-1}) \cdot Y_2 \\ &= X + (1-Z^{-1}) \cdot Q_1 + [-(1-Z^{-1}) \cdot Q_1 + (1-Z^{-1})^3 \cdot Q_2] \\ &= X + (1-Z^{-1})^3 \cdot Q_2 \end{aligned} \qquad (5\text{-}28)$$

Figure 5-11 is the simulation model of two-stage three times noise shaping model. The input is sine wave. Figure 5-12 shows the spectrum in the output. Comparing 5-7(b) and Figure 5-12, we can conclude that, in audible range, the two stage three times noise shaping can achieve the same SNR with the three stages noise shaping model

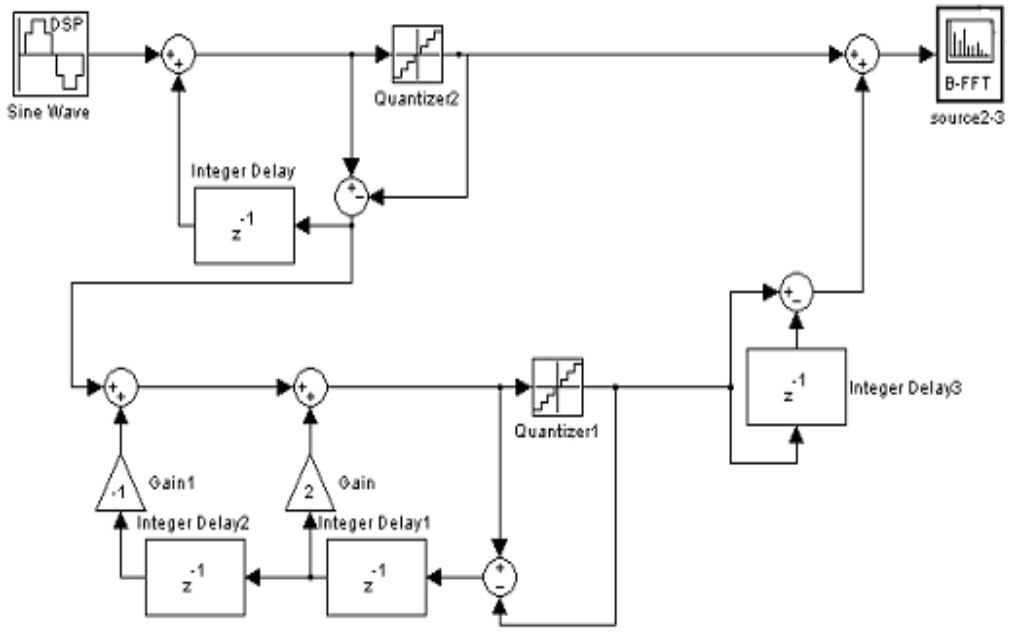


Figure 5-11 Two-stage three times noise shaping model in simulink

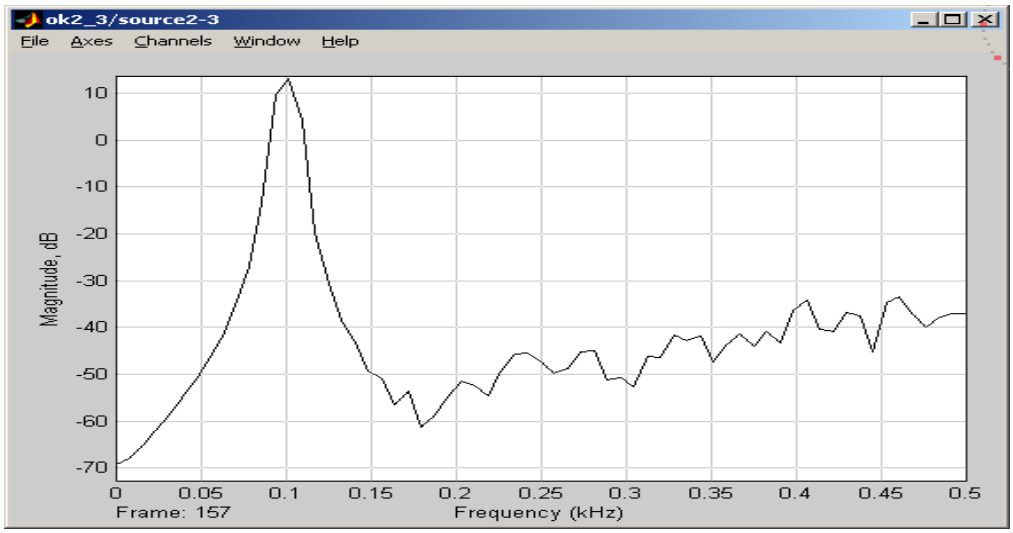
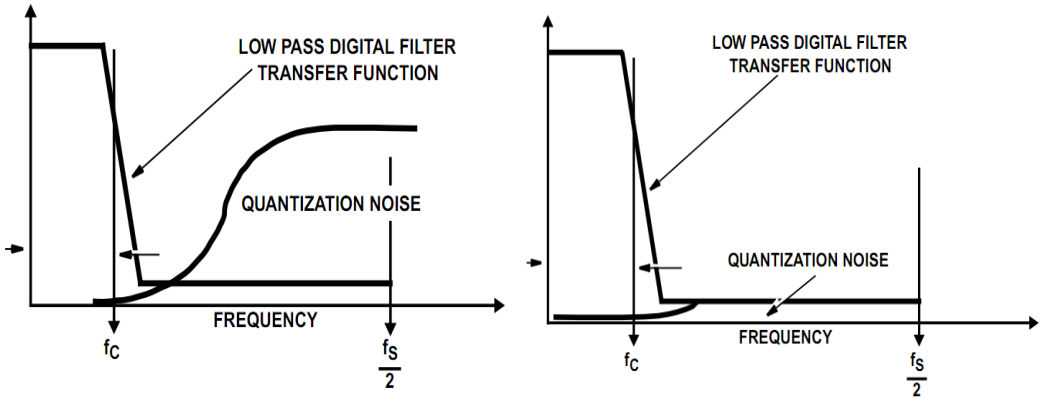


Figure 5-12 The output of two stage three times noise shaping model in simulink

5.6 Post process of low pass filter

After we shape the white noise to higher frequency band, a low pass filter can be used to cutoff the high frequency white noise, as shown below. In this way, the total power of white noise was decreased in the audible frequency band.



6. High-Level Diagram of the System

Please refer to the diagram of the design. For the system, the input signals are:

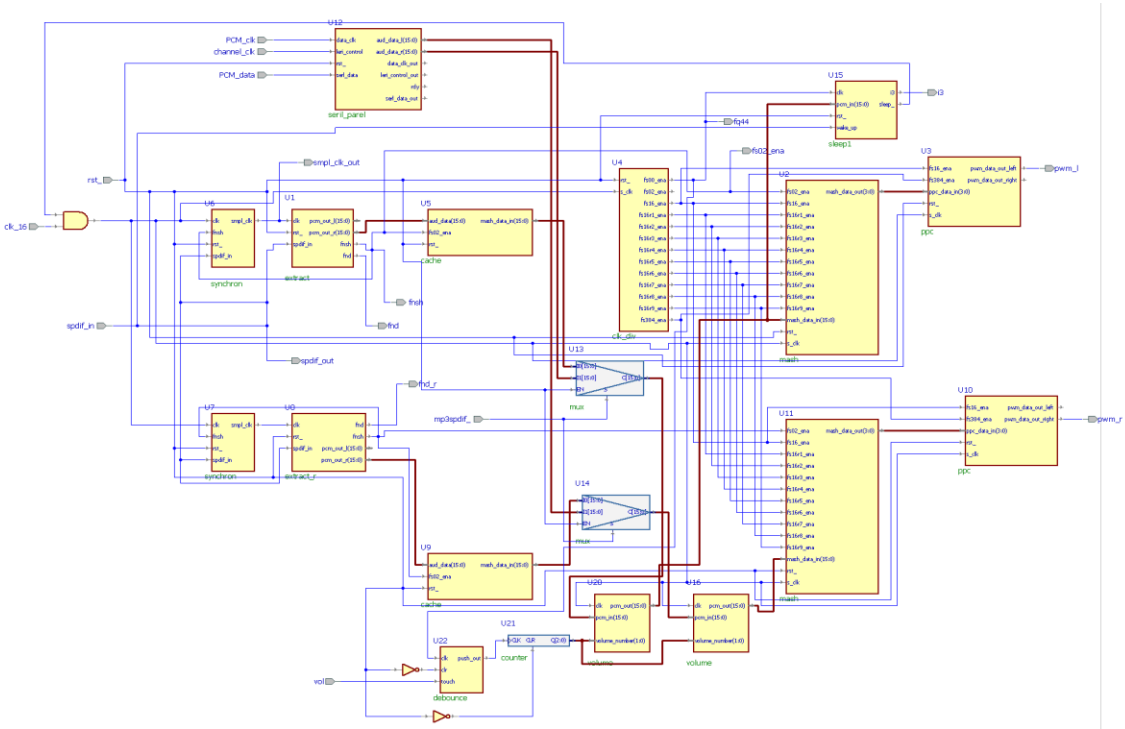
- rst_n: system reset, active low
- clk_16: system clock. It should be 16.9344MHz. We used on-chip PLL to generate.
- spdif_n: digital audio signal from DVD player. It is serial signal which uses SPDIF standard(“IEC 60958)

- vol: music volume control input. Each pulse makes music lauder a little bit. Totally it has 3 levels, 3 pulses will make music go back to original volume value

Output signals:

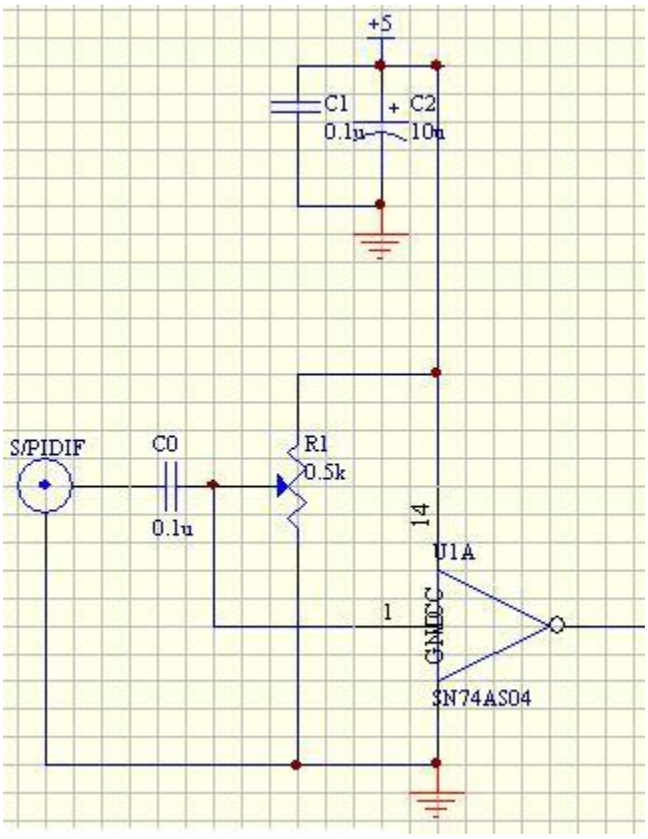
- pwm_l: pwm audio signal, left channel
- pwm_r: pwm audio signal, right channel

Other signals are for debugging.

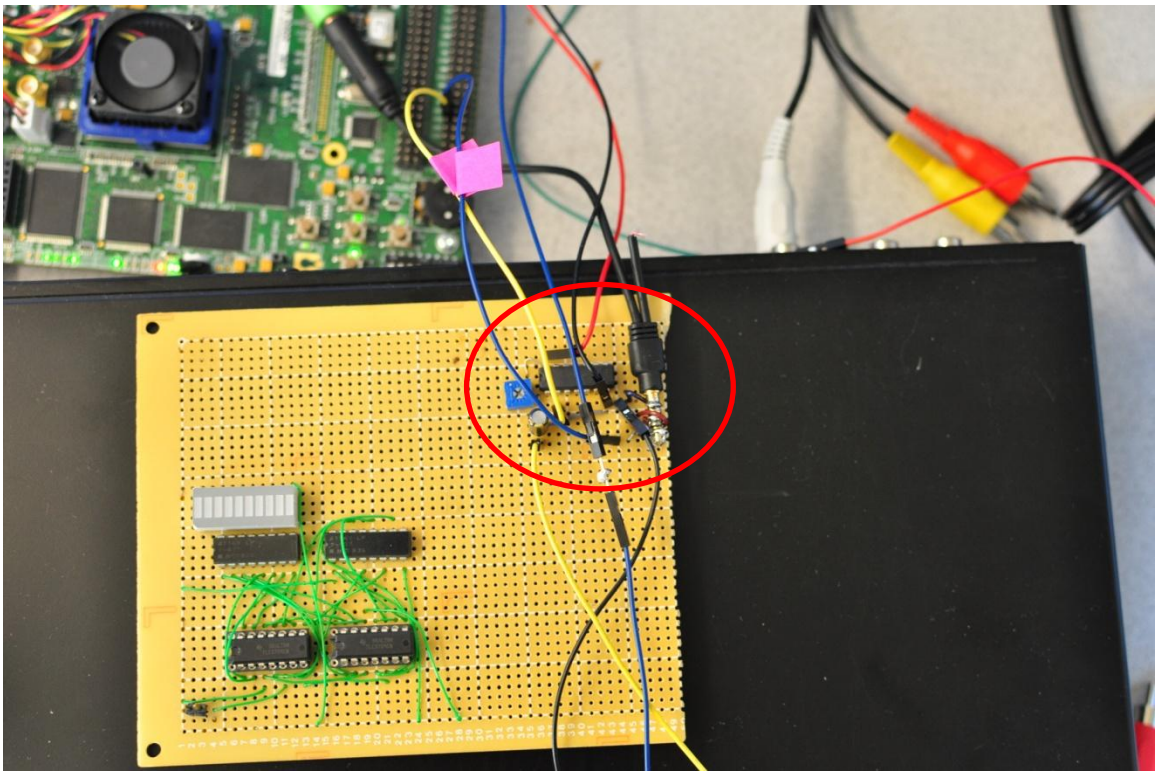


7. Input Signal Voltage converting

Because the serial output signal from DVD player is in the range of -0.5V to 0.5V. But our FPGA works on 0V to 5V. So we need to shift the voltage region of DVD output into 0V to 5V. We used a Schmidt trigger (SN74LS14N) combined with capacitors and rheostat shown below to make it works. In the figure below, the SPDIF is the signal from DVD player, and the output of the Schmidt trigger is what we want.



The real circuit is shown below in red circle:



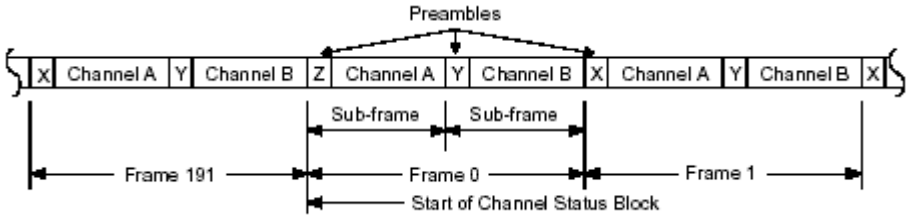
The demo of how this module works:

http://www.youtube.com/watch?v=uBhcLwn46QM&feature=youtu.be

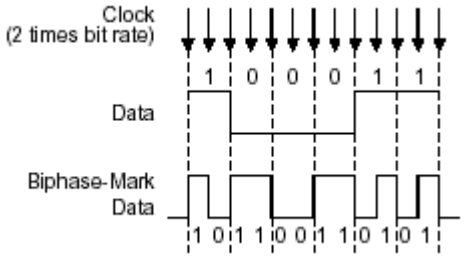
8. Implementation

8.1 Clock Synchronization Module

S/PDIF (Sony/Philips Digital Interface) is a digital audio interconnect used in consumer audio equipment over relatively short distances. S/PDIF is a biphase encoding data. The benefit of biphase encoding is that we can extract clock signal from data signal. In the figure blow, the major unit is a block, and each block has a header Z which contains 192 frames. Each frame contains two sub-frames (one for right channel and the other for left channel). The preamble has three types, X, Y and Z.

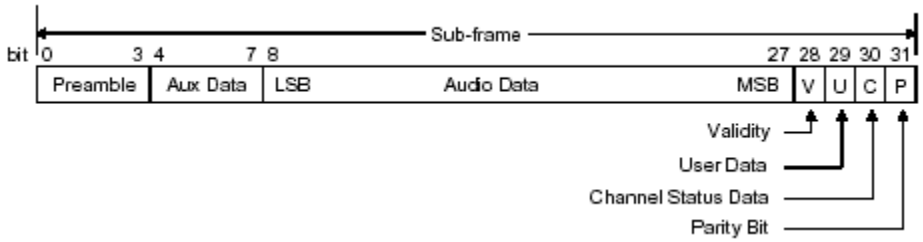


Biphase-Mark: Show on the figure below, the clock frequency is two times of original data bit rate. The original data is encoded into biphase-mark data. In the encoding data, each bit is indicated by two bits. '1' becomes "01" or "10" and '0' becomes "00" or "11". When two of the data are continuous ("11" or "00"), the third one must to change. Therefore, in enodng data, there is no data that contain three continuous '0' or '1'.



Sub-frame: One frame contains two continuous sub-frames. Each sub-frame is a sound data of one channel. One sub-frame contains 32 bits data but this data is encoding by byphase-mark so the total number are 64 bits. In the figure below, 0 to 3 bits indicate the preamble for sub-frame, 4 to 7 are aux data, and 8 to 27 are sound data. We use 16 bits in

PCM so the data we used in sub-frame are 16 bits (12 to 27). From 8 to 31 bits are validity, user data, channel status data and parity bit.

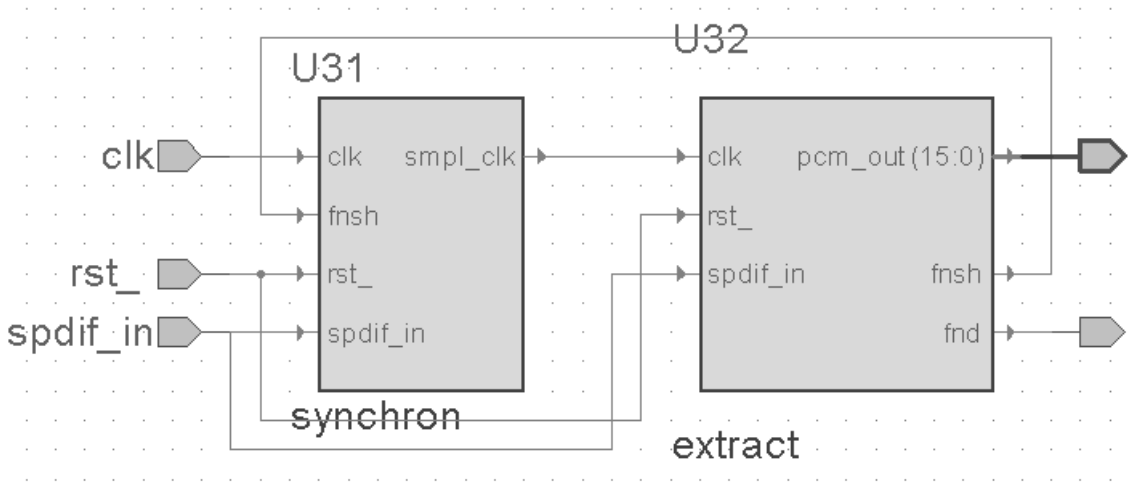


Preamble: The three types of preamble are X, Y and Z. For indicating the data contained in preamble is different from normal data, the preamble contains three continuous ‘1’ or ‘0’. The figure blow shows the biphase patterns for preamble.

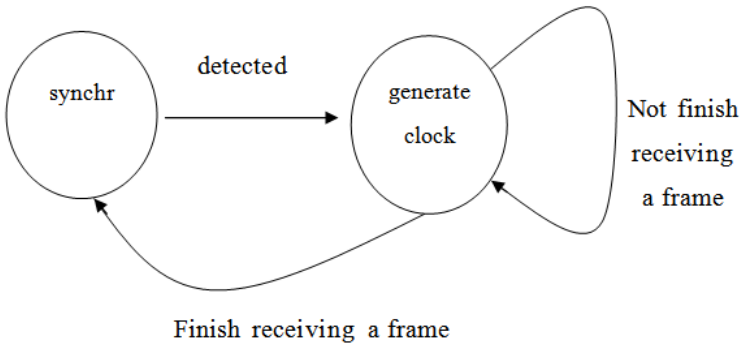
	Biphase Patterns	Channel
X	11100010 or 00011101	Ch. A
Y	11100100 or 00011011	Ch. B
Z	11101000 or 00010111	Ch. A & C.S. Block Start

Clock frequency: In S/PDIF, the clock rate is not fixed. The rate changes by transiting data’s rate. The frequency of data which is stores in CD is 44.1kHz. Every frame has two channel’s data. Every data are 32 bits. And S/PDIF is biphase mark encoding. Therefore, the clock rate is $44.1\text{kHz} \times 2 \times 32 \times 2 = 5.6448\text{MHz}$.

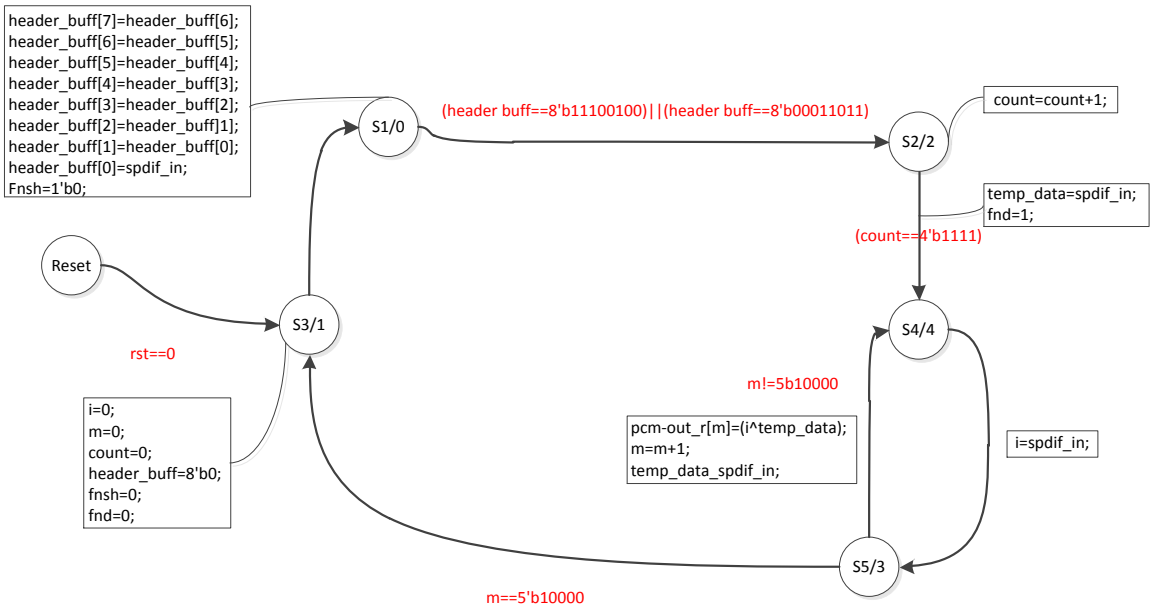
For fetching S/PDIF data, we choose the frequency that is three times of S/PDIF data rate (56488x3 =16.9344). We used two components for data synchronizing and extracting (synchron and extract). The figure blow shows how we connect our design. The synchron produce a clock (which is decided by spdif_in signal) to extract. Then extract can fetch data from spdif_in depends on the input clock rate.



Synchronized Clock: When synchron detect the input data clock rate, this component will produce a suitable clock signal to extract continuously until the extract send frame end signal to synchron.

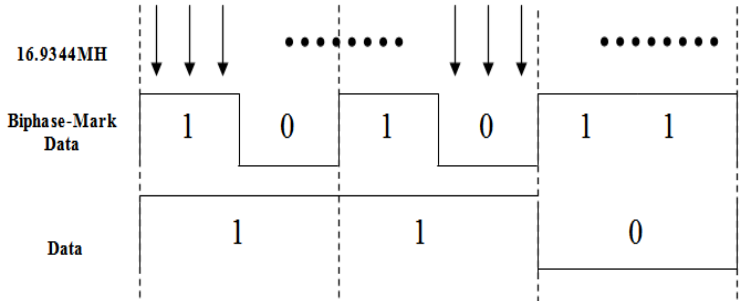


The detailed FSM is shown below

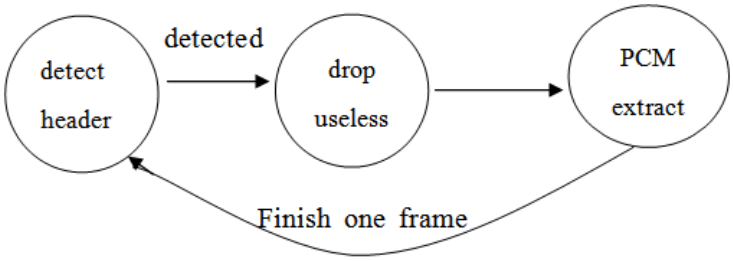


8.2 Data Extraction Module

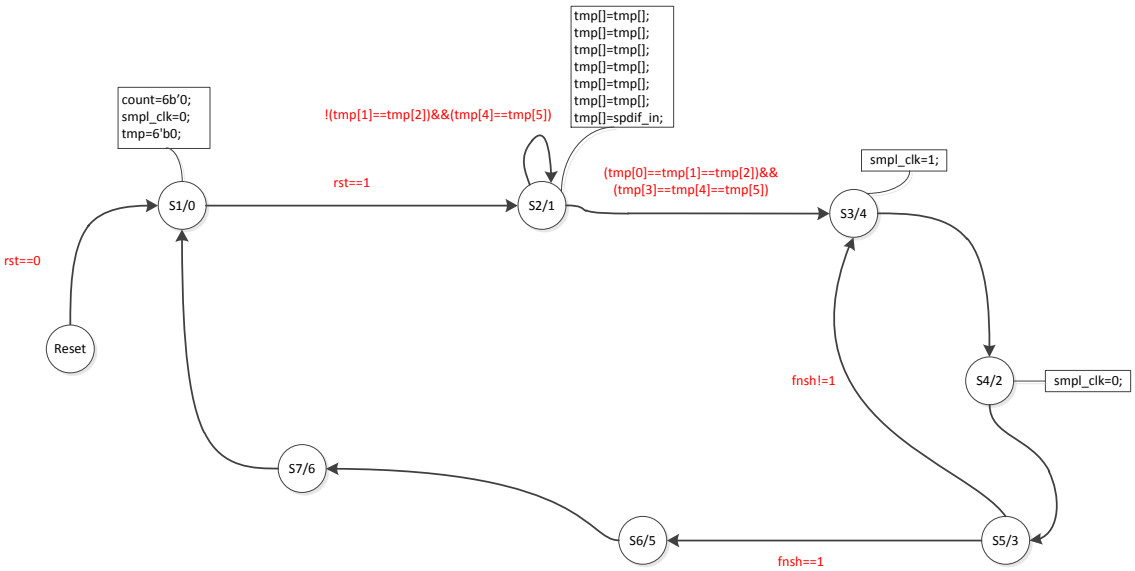
The figure below shows how we get clock from S/PDIF data signal. When our component detect there are two pair of three same continuous bits (e.g. “000111” or “111000”), it will start to produce clock signal.



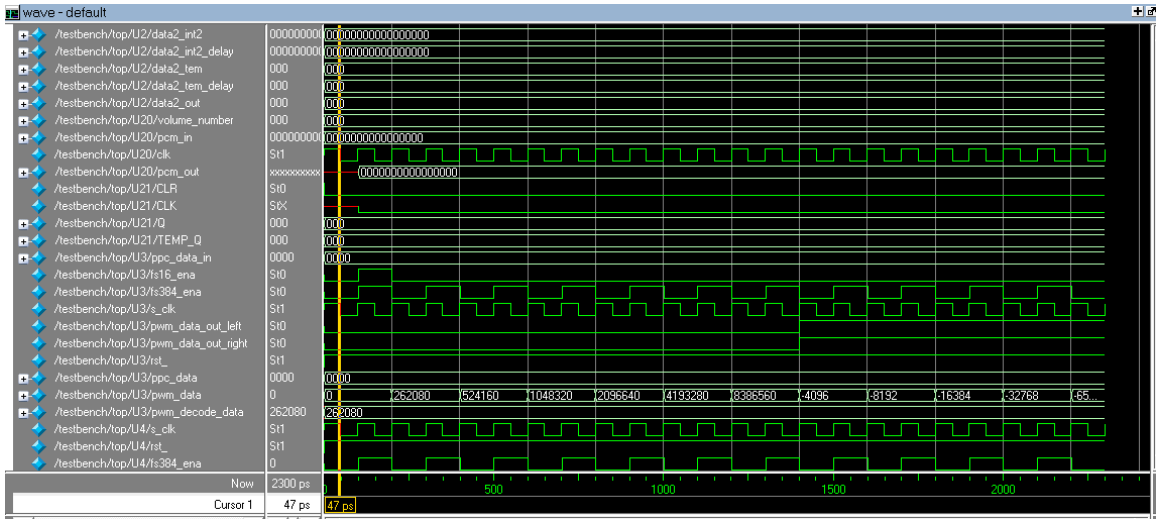
The blow picture shows when extract component detect the preamble (11100010 or 00011101 for first channel; 11100100 or 00011011 for second channel) of one sub-frame, this component will start to count and decide which data are needed to be fetched and which one will be dropped. Then, the extract component will send the data which it fetch to PCM.



The detailed FSM shown below



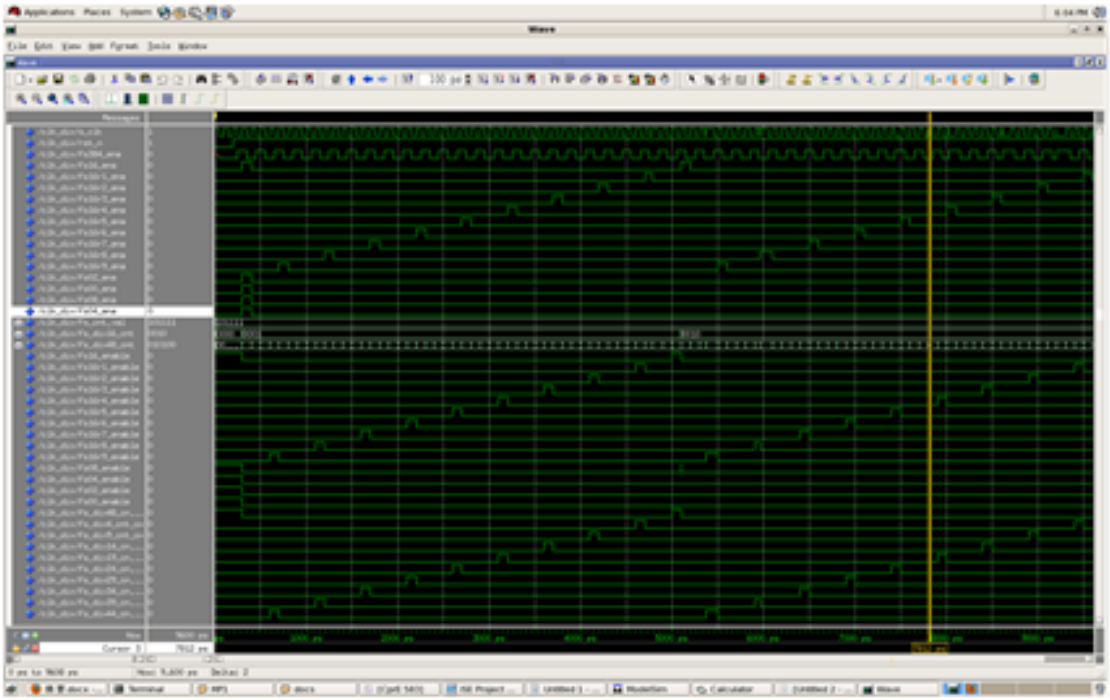
Simulation:



8.3 Clock divider module

Module “clk_div” is a divider, which divides system clock into few inner clocks. After that, each signal is send to module “ppc” and module “mash”, so these two modules help generate eleven different PWN signals.

Communal music players use 16.9344MHz as normal system clock. As we can see on the figure below, module “clk_div” divides the input clock, which is 16.9344 MHz, into many signals:

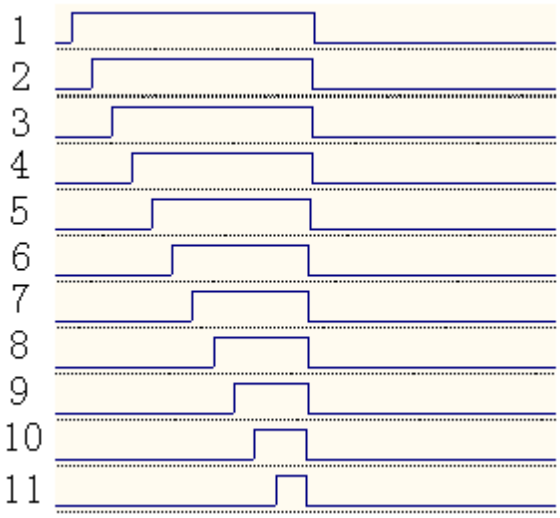


- fs00_en is 44.1kHz, which is used only for test purpose.
- fs02_en is 88.2kHz, which is also for test purpose.
- fs16_en is 16 times larger than 44.1kHz ; it is 705.6kHz
- fs16r1_ena, fs16r2_ena, fs16r3_ena, fs16r4_ena, fs16r5_ena, fs16r6_ena, fs16r7_ena, fs16r8_ena and fs16r9_ena are all 705.6 kHz. However, each signal has equal delay between them.

These signals are necessary for further MASH module computing. When each fs16_ena signal is active, Module ”mash” and Module “ppc” read sixteen bits PCM simple. In other word, fs16r1_ena, fs16r2_ena, fs16r3_ena, fs16r4_ena, fs16r5_ena, fs16r6_ena, fs16r7_ena, fs16r8_ena and fs16r9_ena spark off Module “mash” and “ppc” to sample each 16 bits PCM.

fs384_ena signal, twelve time as fs16_ena signal, sent to PWM from sample module. fs384_ena is used to generate one bit string PWM data from four bits input.

Finally, PWM module can generate 11 PWM sample signals by using all the fs16_ena and fs384_en.

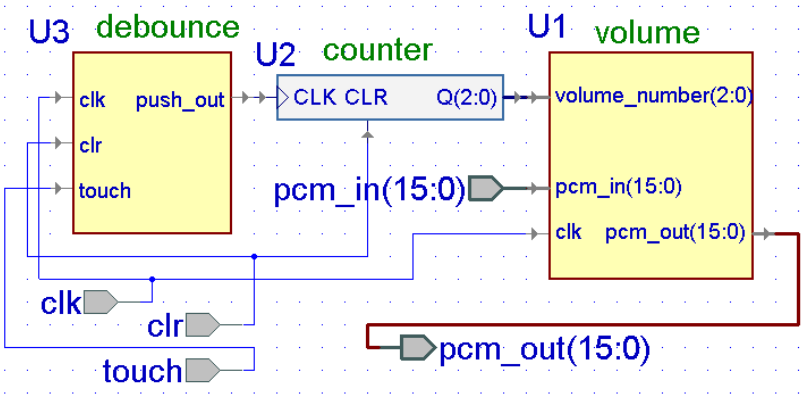


8.4 Sigma-Delta data converting module

For this module, we implement the algorithm mentioned in the chapter 5.5.

8.5 Volume module

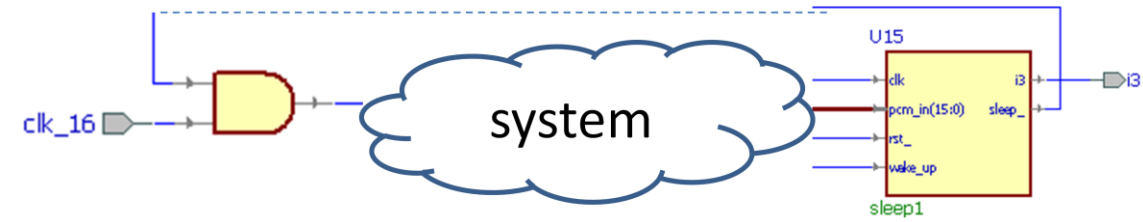
The method of volume adjusting is right and left shifting the input 16-bit PCM vector. For making the volumn lauder, we need to time the 16-bit PCM vector by 2. So we left shift the PCM data. On the other hand, for making the volume lower, we right shift the PCM data, while repeat the MSB.



As shown above, we used a counter to count how many pushes are received, to decide how many times left or right shifting the input PCM data should. However, there is a practical problem, that when we push the button by fingers only one time, there are shakings and multiple pushes are received. So we designed a debounce module which can screen out this kind of glitches.

8.6 Sleep mood module

Figure 2.1 shows the block diagram of sleep module.



The input of sleep module is

- clk

With the stimulation of the clk signal, sleep module inspects the 16-bit audios which are inputs to D/A converter. The system without valid inputs begins clocking when the clk signal begins.

- pcm_in

The 16-bit audios inputs to D/A converter.

- reset_n

Reset signal

- wake_up

When the system goes into sleep mode, once the event of this signal is inspected, this signal is connected to the serial audio input of the system.

The output of sleep module is

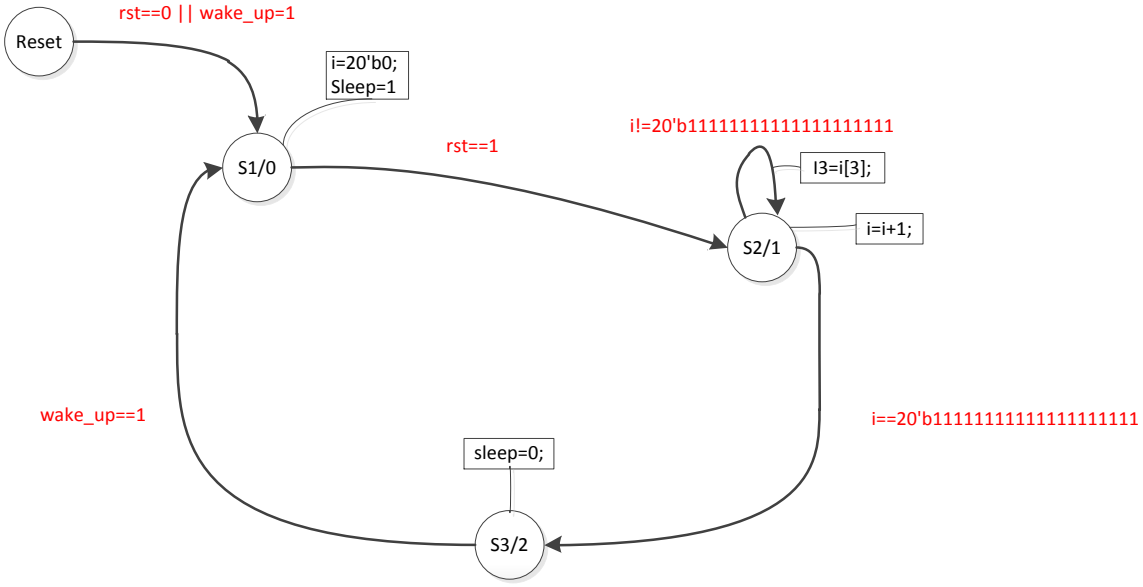
- i3

When there is no valid input in the system, the system begins clocking. This signal is used to inspect if the sleep module has began clocking.

- sleep_n

This signal is used as the power supply of the chip. When the system needs to “sleep”, the output of the signal is logic low, the crystal oscillator stops working. When the system is waked up, the output is logic high, the oscillator begins working.

The state machine of the sleeping module is shown as follows.



There are three states in the sleep module: S1, S2 and S3. When the system resets, the state goes to S1, the counter is cleared to zero, and sleep_n is set to “1”, the crystal oscillator begins to work.

After the reset is done, the state goes to S2. In S2, the system keeps inspecting the input of the whole D/A converter to check if pcm_in is zero. Once the pcm_in is zero, it means the system has new valid input, so the state goes back to S1, the module begin resetting again.

On the other hand, if the counter continues counting, when the counter is full, it goes to S3. The sleep_n is set to “0”, the power supply of crystal oscillator is turn off, the whole system begins sleeping. Then when wake_up signal changes, the sleep module resets

simultaneously and goes back to S1, and then sets sleep_n to “1”, the crystal oscillator is powered again, and wakes up the whole system to begin working.

The demo of this module:

http://www.youtube.com/watch?v=Q74URMVuhF8&feature=youtu.be

We can see, after we unplug the input signal of our system, 3 seconds later, the output of our system shuts down.

9. Conclusion

This project is a very good practice for us to implement an idea using FPGA for what we have learned in CPRE583. And it is also a good training for us to integrate and debug a real system. For example, in the sleep mode module, we need to design a debounce module to screen the finger shaking and to suppress generated glitches when we push the button on the FPGA board. We successfully finished this project and we learn much.

10. reference

http://en.wikipedia.org/wiki/Delta-sigma_modulation

http://www.cscamm.umd.edu/programs/ocq05/adams/adams_ocq05.pdf

http://www.beis.de/Elektronik/DeltaSigma/DeltaSigma.html

http://www.intersil.com/data/an/an9504.pdf

http://hephaestusaudio.com/media/2009/07/MASH-Delta-Sigma.pdf