Re: DSF2FLAC

Jagger Huang to: Jim Yam 2018/12/28 15:23

Dear Jim,

Thanks for your information.

In fact the DSD2PCM library uses the same optimized processing as DSF2FLAC does(dsd2pcm.c , line 126). Use a 2nd half (48 coeffs) of a 96-tap symmetric lowpass filter to generate a lookup table,avoiding

the multiplication operate in the decoded loop. And its fir filter pre-calculated looktable takes 6 kbytes in the static memory. But the DSD2PCM only support 352khz, for other sample rate it needs a SRC

or a redesigned fir filter. I find that DSD2FLAC library contains the preset processing generates the f64(DSD64) -> 352,186,88.2 khz lookup table.(dsd_decimator.cpp line 57) It is very useful if we want to

do the same convertion.

dsd2pcm.c

With best regards,
Jagger Huang/黄剑钢.

Tel: (86)755-82463828 ext.323
E-mail Address: jagger.huang@gpe-hkg.com
Address:
16F, C Building, Oriental Xintiandi Plaza,1003 Shen Nan Avenue,
Futian District Shenzhen, PRC
深圳市福田区深南大道1003号东方新天地广场C栋16楼 邮编:518043

Jim Yam Dear Jagger, For your information, the fo... 2018/12/28 14:21:47

From: Jim Yam/SAE/GPESZ/GPE
To: Jagger Huang/SAE/GPESZ/GPE@GOLDPEAK,
Date: 2018/12/28 14:21
Subject: DSF2FLAC

Dear Jagger,

For your information, the following email from Jack explains a faster algorithm to convert DSD to PCM. The method is used in the DSF2FLAC project.

The link for the source codes of DSF2FLAC is as follows:

https://github.com/hank/dsf2flac

The file "dsd_decimator.cpp" is the main routines for the conversion in the dsf2flac project.

With best regards,

Jim.

----- Forwarded by Jim Yam/SAE/GPESZ/GPE on 2018/12/28 14:14 -----

From: Jack Oclee-Brown <jack.oclee-brown@kef.com>
To: "Jim, Yam" <jim.yam@gpe-hkg.com>,
Cc: "Nick, Lam" <nick.lam@gpe-hkg.com>, "Solong, Zeng" <solong.zeng@gpe-hkg.com>
Date: 2018/12/24 19:40
Subject: Re: MQA on StreamUnlimited

Hi Jim,

This is a good question!

There is a shortcut that you can use for the FIR low pass filter for DSD to PCM conversion that increases the efficiency substantially compared to a standard FIR filter.

Start with a normal FIR filter:

$$Y[n]=\sum_{i=0}^{N} b_i \cdot x[n-i]$$

Now split the summation into two summations where the inner summation is always from 0 to 7 (note N must be a multiple of 8 but can be zero padded to achieve this).

$$Y[n]=\sum_{i=0}^{N/8}\sum_{j=0}^{7} b_{(8i+j)} \cdot x[n-8i-j]$$

With DSD data the value of x[n] can only be 0 or 1 so this means there are only 2^8 = 256 possible patterns for $[x[n-4i-0],x[n-4i-1],x[n-4i-2],x[n-4i-3],x[n-4i-4],x[n-4i-5],x[n-4i-6],x[n-4i-7]]$:

Pattern0 = [0 0 0 0 0 0 0 0]
Pattern1 = [0 0 0 0 0 0 0 1]

Pattern2 = [0 0 0 0 0 0 1 0]
Pattern3 = [0 0 0 0 0 0 1 1] etc....

So we can precompute the result of the inner summation for every possible pattern and store the result in a lookup table (see dsdp_decimator.cpp attached line 117)

The table size is N/8 by 256.

After the table is computed then the filtering process is just the outer summation. So it's only N/8 additions for each output sample. (see dsdp_decimator.cpp line 195). The efficiency is substantially higher.

normal FIR polyphase: 50,803,200 macs
fast method polyphase: 6,328,350 additions.

There is another benefit. We can do the multiplications for the lookup table values at a higher resolution (e.g. 64bit floating point) and then decimate to a lower resolution for the outer summation loop calculation (e.g. 32bit floating point). I'm sure that 32bit floating point will be high enough resolution for the outer summation loop.

It's possible to increase the efficiency further by increasing the summation in the inner loop from 8 to 12 or 16. But then we need a bigger lookup table. Also note that the maximum output sample rate that we can support is related to the inner loop size:

Max output rate with 8 samples in inner loop: 44100*64/8 = 352800Hz
Max output rate with 8 samples in inner loop: 44100*64/12 = 235200Hz
Max output rate with 8 samples in inner loop: 44100*64/12 = 176400Hz

The output from the XMOS must always be 96kHz or 192kHz. I propose that we always convert DSD at a fixed rate (e.g. 352800Hz or 176400Hz) and then we use the XMOS sample rate converter to compute a 96kHz or 192kHz output (depending on if we have cable or wireless master to slave connection).

At this moment I don't think we should allow the user to choose the filter type.... but it's an interesting idea. It might be interesting for the user but we already have many settings that they can change so I'm not certain we should add more. What's your opinion?

If we plan to have a fixed output resolution from the DSD->PCM calculation then we will need two filters if we support DSD64 and DSD128 (not confirmed yet).

Let me know if you have any questions or concerns.

Kind regards,

Jack.

On 24/12/2018 06:28, jim.yam@gpe-hkg.com wrote:

> Hi Jack,
>
> In your new proposal, you propose to decode the DOP2PCM in XMOS. However, I think XMOS may not be fast enough to do the decoding because the computation is intensive.
>
> I have a quick look of the source codes used in DSD2FLAC. DSD2FLAC uses a 575-tap FIR filter to convert the DSD64 bit stream to 88.2kHz PCM. The filter coefficients are 64-bit double precision floating numbers. The number of floating point multiplications is 50,803,200~=50MIPs even if the Polyphase filter realization is used.
>
> If we need to convert DSD128 bit stream to 88.2kHz PCM, the number of floating point multiplications will be higher because the sampling rate is higher and a longer FIR filter has to be used.
>
> However, the computations will be lower when DSD64 bit stream is converted to 176.4kHz or 352.8kHz. You may want to convert 352.8kHz PCM to 96kHz for later DSP processing.
>
> Do you know if there are other faster methods for converting DSD to PCM? Do you think that if 32-bit fixed-point FIR filtering is accurate enough for the product? By the way, will you let the user to choose different filters for listening?
>
> Thanks and best regards,
>
> Jim.
>
> [attachment "Active Floorstander Proposal v014.png" deleted by Jagger Huang/SAE/GPESZ/GPE]