Lab Project: Two-channel Filterbanks on UniDAQ2 board [1]

Student group:	Professors comment:	Record editor:
Date:		Group members:
Professor:		

1 Objectives

The purpose of this lab is to understand the UniDAQ2 board and its facilities to implement filter-banks with a decimation/interpolation factor **L=M=2**. The UniDAQ2 board uses a Texas Instruments TMS320C6747 DSP with floating-point support. We will however use fixed point representation with 16 bit only because most implementations are done with fixed-point arithmetic due to cost reasons.

After this laboratory you should

- be able to design a digital filter for a filterbank with (small) amplitude distortions and with perfect reconstruction
- be able to write the MATLAB code to design and simulate a filterbank in the time domain
- be able to write the ANSI C code for a filterbank and implement it on the hardware
- be able to operate the UniDAQ2 board with two timers having <u>different</u> settings such that the ADCs and DACs operate at the <u>different sampling</u> frequency ($F_{s1} \neq F_{s2}$)
- be able to measure the amplitude response using an UPV Audio Analyser and understand the amplitude spectra measured using an FFT

2.1 Abbreviations

ADC	Analogue-to-Digital converter
CCS	Code Composer Studio (currently ver. 12 is used)
CODEC	A chip containing an ADC and a DAC
DAC	Digital-to-Analogue converter
R&S UPV	An audio spectrum analyzer made by Rohde & Schwarz
SRC	Sample Rate Converter
TI	Texas Instruments
UniDAQ2 board	A versatile data acquisition and signal processing system for industrial applications, scientific research and education from the company d.signT

2.2 Material for the lab

1. You do not have to set up a new project for the implementation. As known from the tutorial, a UniDAQ2 project is available in **d:\ti_work**.
2. Some helpful M-files like inputs(..), write_coeff(..) can be found in MOODLE, see <u>MES_MATLAB_Lecture_Examples</u>
3. Use Getting Started again for handling UniDAQ2 board and CCS.

3 *Lab task descriptions*

3.1 Filter-banks with FIR filters having amplitude distortions

Write a MATLAB script **`filterbank_FIR_amp_dist.m`** which allows to design an FIR filter $H_0(z)$ (in polyphase structure, M=2) for a filterbank, as shown in Fig. 3.1.1.

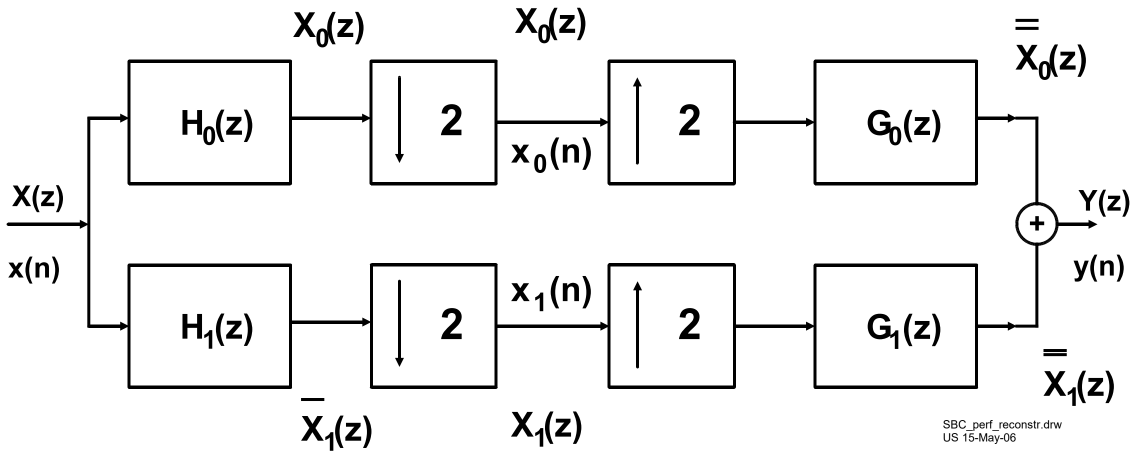

Fig. 3.1.1 Arrangement for filterbank with amplitude errors

This FIR Filter has the following specification.

* F_{s1}= 50 kHz
* pass-band edge frequency 8900 Hz,
* stop-band edge frequency 16100 Hz (⇔ ($F_{s1}/2$ - f_{pass})),
* pass-band ripple 1.45e-4 (deviation from 1, value corresponds to δ_p, so this value is <u>not</u> in dB !)
* minimum stop-band attenuation 40 dB (thus ripple 0.01)

Polyphase decomposition allows to implement the low-pass and high-pass filters with a cross-over network, see below and next page. This way, the computational effort is reduced dramatically.

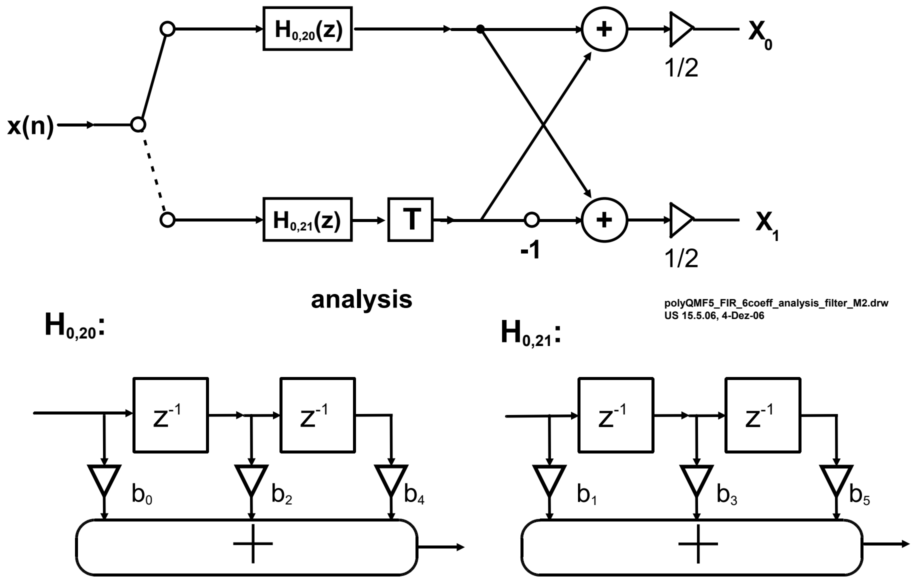

Fig. 3.1.2a Signal-flow diagram, analysis side for efficient implementation

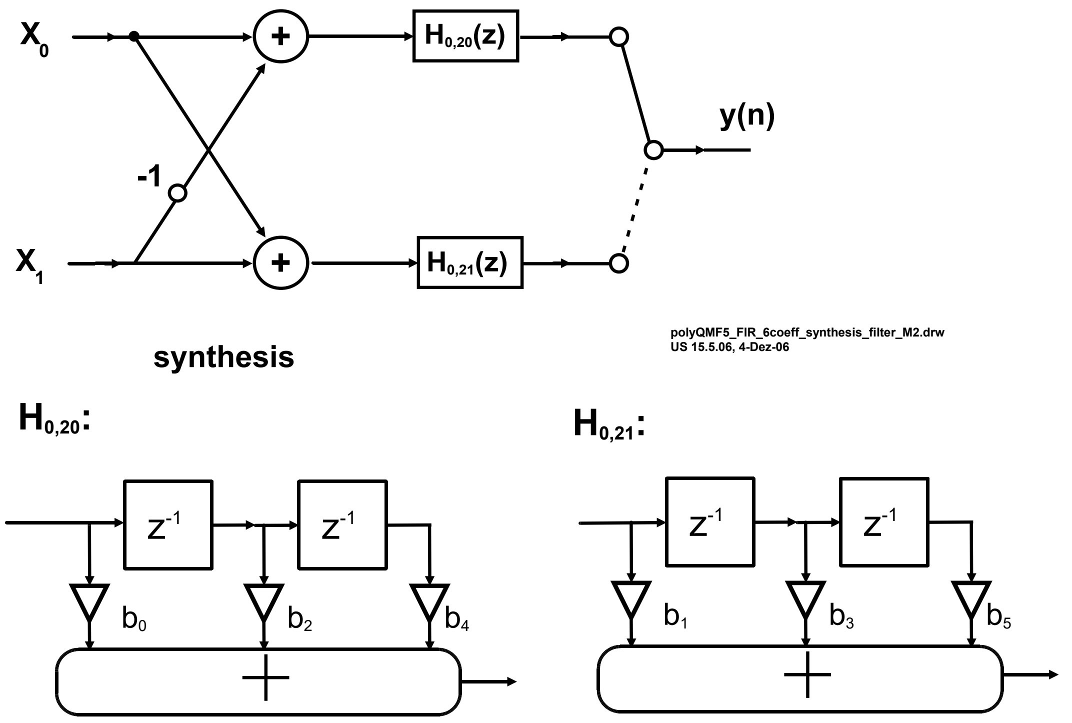

Fig. 3.1.2b Signal-flow diagram, synthesis side for efficient implementation

The number of coefficients must be even ! Design the filter using MATLAB and show the amplitude response and the phase response of this filter (MATLAB results) in ***Attachment A.*** Calculate the overall amplitude response H(z) = Y(z)/X(z) using MATLAB and show the plot in dB as ***Attachment B.***

Store the variables for the coefficients and the delays as well as the coefficient values in a file **`filterbank_FIR_amp_dist.h`**. You may generate this file using the function **`write_coeff.m`**. Show your MATLAB code of **`filterbank_FIR_amp_dist.m`** also in ***Attachment C.***

Create a CCS project **`filterbank_FIR_amp_dist`** and develop the program code for the filterbank. Make sure that the polyphase components are realized by using the assembly module for the FIR filter, **`fir_filter_asm_sc.asm`**, as done in the previous labs.

```
short FIR_filter_sc( short FIR_delays[], short FIR_coe[], short int
N_delays, short x_n, short SHIFT);
```

You also have to make sure that the two timers of the DSP are set to the correct sampling frequencies. In main( ), LL and MM must therefore be set correctly.

Show the signal flow diagram in ***Attachment D*** and your complete ANSI C source code (main( ) and INT routines) in ***Attachment E.***

Measure the amplitude response from the input to the output using **`filterbank_FIR_amp_dist_in_out.set`** and show the plot in ***Attachment F.***
Measure the amplitude response from the input to the lower and higher sub-band output using **`filterbank_FIR_amp_dist_subbands.set`** and show the two plots in ***Attachment G.***

Change (on screen 1, most-right window) the UPV analyzer settings from “RMS” to “RMS selective”. Answer in ***Attachment H*** : Why does the high-pass filter carrying the upper sub-band disappear from the UPV screen ?

Modify the input frequency of the UPV generator around 1000 Hz (using the knob on the UPV analyzer) and carry out an FFT in the frequency range 20 Hz to F_s of the output sequence (use set file **`filterbank_FIR_amp_dist_FFT.set`**).

Show one plot of the FFT for an input frequency of 1000 Hz in ***Attachment L (Attachments I, J and K do not exist)***.

Explain: Why do you see several “lines” on the FFT plot?

3.2 *Filter-banks with FIR filters having perfect reconstruction properties*

An arrangement for perfect reconstruction is shown in Fig. 3.3.1.

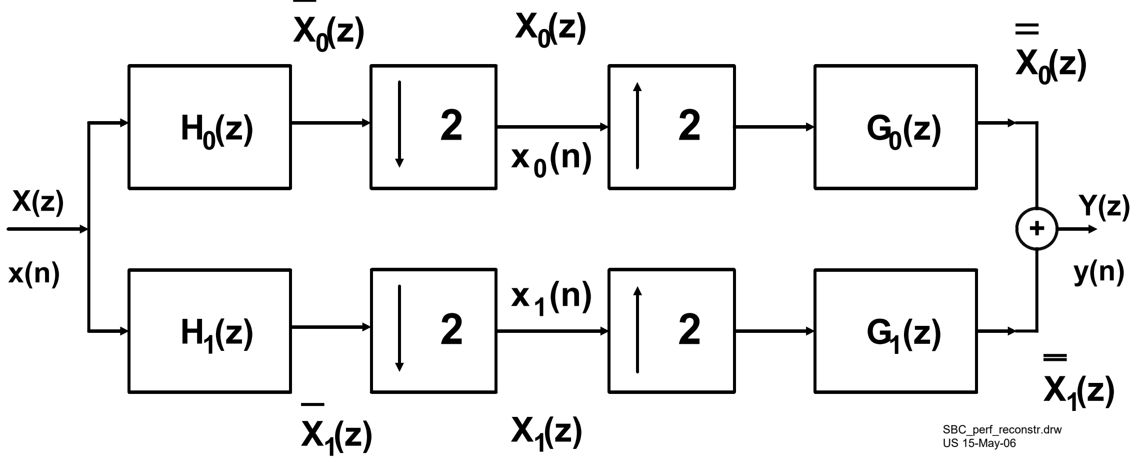

Fig. 3.2.1 Arrangement for perfect reconstruction (same Fig. as Fig. 3.1.1, but different filters used!)

$H_0(z)$ is realized by using $H_0(z) = H_{0\_20}(z^2) + z^{-1} H_{0\_21}(z^2)$, thus by using a polyphase decomposition. The same principle can be applied for $H_1(z)$, $G_0(z)$, and $G_1(z)$. Let $H_0(z)$ be the prototype low-pass filter.

The impulse responses are then as follows (shown for 4 coefficients only):

$$h_0(n) \text{ as designed with MATLAB}, \Rightarrow H_0(z) = a_0 + a_1 \cdot z^{-1} + a_2 \cdot z^{-2} + a_3 \cdot z^{-3}$$

$$h_1(n) = h_0(N-1-n) \cdot (-1)^n \Rightarrow H_1(z) = a_3 - a_2 \cdot z^{-1} + a_1 \cdot z^{-2} - a_0 \cdot z^{-3}$$
$$g_0(n) = h_0(N-1-n) \cdot 2 \Rightarrow G_0(z) = 2 \cdot [a_3 + a_2 \cdot z^{-1} + a_1 \cdot z^{-2} + a_0 \cdot z^{-3}]$$

$$g_1(n) = -h_0(n) \cdot (-1)^n \cdot 2 \Rightarrow G_1(z) = 2 \cdot [-a_0 + a_1 \cdot z^{-1} - a_2 \cdot z^{-2} + a_3 \cdot z^{-3}]$$

Thus, altogether, there are **four polyphase filters (i.e. eight polyphase branches)** involved, two for decimation and two for interpolation.

Accordingly, assuming that $h_0(n)$ has for four coefficients $[a_0, a_1, a_2, a_3]$, the polyphase component $h_{0\_20}(n)$ is thus represented by $[a_0, a_2]$, $h_{0\_21}(n)$ by $[a_1, a_3]$, $h_{1\_20}(n)$ by $[a_3, a_1]$ and $h_{1\_21}(n)$ by $[-a_2, -a_0]$. Similar results can be obtained for the polyphase representation of g_0 and g_1.

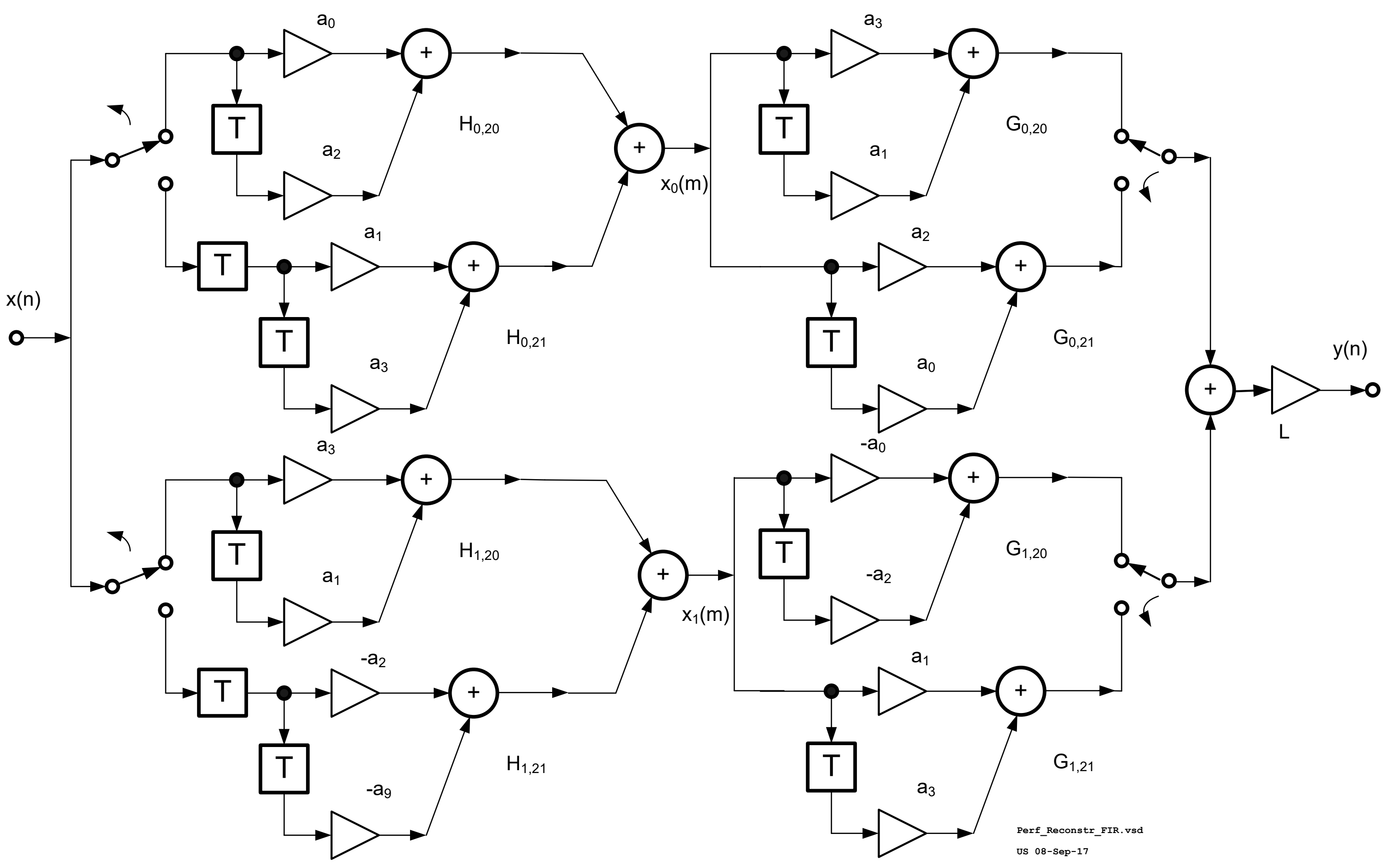

Fig. 3.2.2 Arrangement for perfect reconstruction (same Fig. as Fig. 3.1.1, but different filters used!)

Note that in this case four <u>different</u> filters are used. Therefore, the realization using a cross-over network is not possible.

The computations of the coefficients have already been carried out. Each branch contains exactly 32 coefficients. **The following header file is available in MOODLE.**

```
//---------------------------------------------
// designed with -- perf_reconstruction.m --
// File : perf_reconstruction.h
// Fs_in = 33333.0 Hz
// fpass =   0.45 * Fs_in
// fstop =   0.55 * Fs_in
// delta_pass (most be equal to dBstop !!!) = 1.00e-004
// delta_stop_dB = -80.00
// N_FIR = 126
//---------------------------------------------
#define N 32
short h0_20[]={
   3748,  14930,  -2002,   -838,   1538,  -1633,
   1535,  -1379,   1211,  -1050,    901,   -765,
    646,   -537,    445,   -357,    293,   -226,
    177,   -134,     98,    -70,     48,    -31,
     18,     -9,      3,      1,     -3,      3,
     -3,      6,};
short h0_21[]={
  10926,   9091,  -6297,   4210,  -2938,   2147,
  -1634,   1287,  -1042,    862,   -726,    617,
   -531,    456,   -396,    335,   -293,    246,
   -207,    174,   -142,    115,    -92,     71,
    -54,     40,    -29,     20,    -13,      8,
     -6,     -2,};
short h1_20[]={
     -2,     -6,      8,    -13,     20,    -29,
     40,    -54,     71,    -92,    115,   -142,
    173,   -209,    244,   -294,    334,   -400,
    452,   -533,    617,   -726,    862,  -1042,
   1287,  -1634,   2147,  -2938,   4210,  -6297,
   9091,  10926,};
short h1_21[]={
     -6,      3,     -3,      3,     -1,     -3,
      9,    -18,     31,    -48,     70,    -98,
    133,   -178,    225,   -294,    355,   -449,
    534,   -647,    765,   -901,   1050,  -1211,
   1379,  -1535,   1633,  -1538,    838,   2002,
 -14930,  -3748,};
short g0_20[]={
     -2,     -6,      8,    -13,     20,    -29,
     40,    -54,     71,    -92,    115,   -142,
    174,   -207,    246,   -293,    335,   -396,
    456,   -531,    617,   -726,    862,  -1042,
   1287,  -1634,   2147,  -2938,   4210,  -6297,
   9091,  10926,};
short g0_21[]={
      6,     -3,      3,     -3,      1,      3,
     -9,     18,    -31,     48,    -70,     98,
   -134,    177,   -226,    293,   -357,    445,
   -537,    646,   -765,    901,  -1050,   1211,
  -1379,   1535,  -1633,   1538,   -838,  -2002,
  14930,   3748,};
short g1_20[]={
  -3748, -14930,   2002,    838,  -1538,   1633,
  -1535,   1379,  -1211,   1050,   -901,    765,
   -647,    534,   -449,    355,   -294,    225,
   -178,    133,    -98,     70,    -48,     31,
    -18,      9,     -3,     -1,      3,     -3,
      3,     -6,};
short g1_21[]={
  10926,   9091,  -6297,   4210,  -2938,   2147,
  -1634,   1287,  -1042,    862,   -726,    617,
   -533,    452,   -400,    334,   -294,    244,
   -209,    173,   -142,    115,    -92,     71,
    -54,     40,    -29,     20,    -13,      8,
     -6,     -2,};
```

Write a MATLAB script **`perf_reconstruction.m`** which carries out the following tasks:

- Generate from the coefficients given above a plot of the amplitude responses of $H_0(z)$, $H_1(z)$, $G_0(z)$, $G_1(z)$ for $z=e^{j\omega T}$.
- Plot the overall **amplitude** response from the input to the output of the arrangement in Fig. 3.3.1 and show that it is (almost) flat.
- Plot the overall **phase** response from the input to the output of the arrangement and show that is (almost) linear.

Show the plots and the MATLAB script in ***Attachment M.***

Create a C-file **Filterbank_Perf_Reconstruction.c . Exclude the old C-file from the project and use this one instead.** Write the INT routines in order to implement this perfect reconstruction filterbank on the TI DUETT board. Show the interrupt routines in ***Attachment N.***

Measure the amplitude response from the input to the output of the arrangement in Fig. 3.3.1 using the UPV analyzer (**`filterbank_perf_reconst_in_out.set`**) and show the plot in ***Attachment O.***

Measure the amplitude response from the input to the lower and higher subband outputs of the arrangement in Fig. 3.3.1 using the UPV analyzer (**`filterbank_perf_reconst_subbands.set`**) and show the plots in ***Attachment P.***

Answer in ***Attachment Q:***
How did you check whether the reconstruction is really perfect, i.e., that $y(n) = A \cdot x(n-N)$?