

U3 FIR Hochpass Filter

Ziel

Diese Übung hat zum Ziel ein FIR-Filter mit gegebenen Koeffizienten als Funktion in Hochsprache direkt zu implementieren.

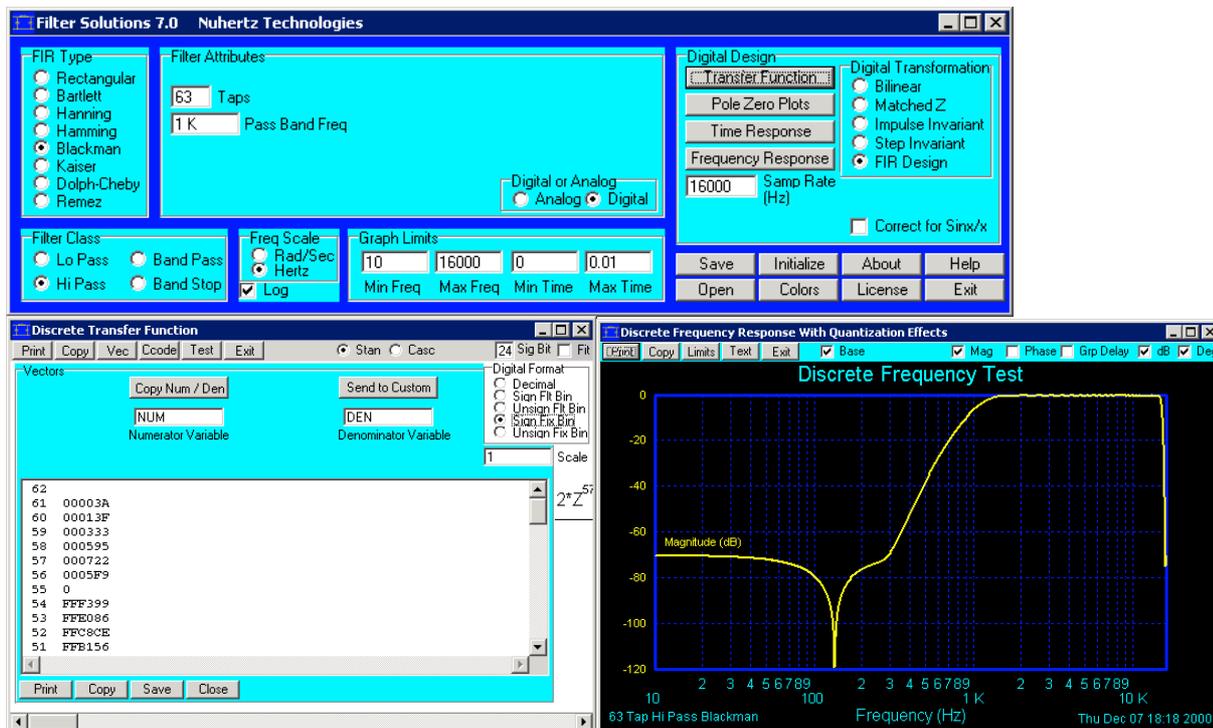
Umfeld

FIR Filter sind einfache und gut zu implementierende Digitalfilter. Da sie nicht rekursiv sind, ist die Stabilität in jedem Fall garantiert. Nachteilig ist die aufwendige Bestimmung der Filterkoeffizienten, die meist eines Iterationprozesses bedarf. In der Praxis wird dies aber durch ein Filtersyntheseprogramm abgenommen.

Die Implementierung der Koeffizienten bedarf je nach Signalprozessor einer Quantisierung, die einen zusätzlichen kleinen Fehler einbringt. Die Rechnung ist grundsätzlich direkt in der Hochsprache möglich, aus Effizienzgründen wird man aber eine Assemblerlösung vorziehen, da nur so die Paralleloperationen des Prozessors optimal genutzt werden können.

Der FIR-Filterentwurf erfolgt nach den Vorgaben:

Hochpass, Grenzfrequenz = 1kHz, Filterlänge = 63 Taps, Samplefrequenz=16kHz:



Die ausgewiesene Übertragungsfunktion (Transfer Function) zeigt den Amplitudengang mit einer Präzision von 24 Bit gerechnet. Die maximal mögliche Sperrdämpfung von 74dB beim Blackman-Fenster wird fast erreicht. Daher wird dieser Entwurf direkt implementiert. Die geforderte Grenzfrequenz wird mit 1.11kHz etwas verfehlt.

Aufgaben

1. Aufsetzen eines neuen, leeren C-Projektes (z.B. Uebung3_FIR_HP1kHz, keine Leerschläge erlaubt) auf der Grundlage des Prototypen-Files. Kommentieren und Anpassen des Prototypen. Vgl. hierzu auch Übung 1.

2. Implementieren der Übertragungsfunktion durch Definition der Filterfunktion, die die eigentliche Übertragungsfunktion aufnimmt:

```
void filterFIR(void)
{
    ....
    y= ....;   Ausgangssignal retournieren
}
```

Die Werte werden mittels Seiteneffekt (d.h. globale Variablen) übergeben.

Die Filterkoeffizienten sind als initialisiertes Array ausserhalb und vor der Filterfunktion zu definieren. Ebenso der History-Buffer zur Aufnahme der $x(n-k)$.

```
....
....
_X _fract delay[63];           // 63 Speicherzellen fuer History und Eingangssignal
/* FIR Filterkoeffizienten, erzeugt mit Filter Solutions 7.0
   FIR HP 63 Taps, fc=1kHz, fs=16kHz
   7.12.2000, G. Krucker
*/
_X _fract znum[62] = {
0.0,
6.96045e-06,
3.80444e-05,
9.76695e-05,
1.70332e-04,
....
9.76695e-05,
3.80444e-05,
6.96045e-06
};
```

Hinweise:

Benutzen Sie zur Rechnung den Datentyp `_fract`. Er verkörpert den Festkommatentyp des DSP56002 Signalprozessors mit dem Wertebereich $[-1,1)$. Zu beachten ist, dass der Codec mit 16kHz Samplerate initialisiert wird. Dazu ist die symbolische Konstante `SAMPLE_RATE_16` aus `CodecConst.h` zu benutzen.

Die Filterkoeffizienten sind als File `FIR_HPCOE1KHZ.C` ab Diskette direkt einlesbar.

3. Austesten mit Musik, evtl. ausmessen des Amplitudenganges mit NF-Voltmeter.

Als Hilfe erhalten Sie einen Auszug einer möglichen Lösung:

```
TASKING EDE [ DSP5600x - S:\E97\DSP\DSPuebungen\Uebung3_FIR_Hochpass\FIR_HP1kHz.pjt ]
Edit Search Project Text Document Tools Window Help EDE
S:\E97\DSP\DSPuebungen\Uebung3_FIR_Hochpass\FIR_HP1kHz.c
9.76695e-05,
3.80444e-05,
6.96045e-06
};

/* Funktionsprototypen */
void codecInit(int, int, int, int); // CS4215 Codec und zugehoerige Interruptroutinen initialisieren
void filterFIR(void);

/* Einfache, aber ineffiziente FIR Filter Routine */
void filterFIR(void)
{ int i;
  y=0.0;
  for (i=0;i<=61;i++)
  { delay[i] = delay[i+1];
    y += delay[i]*num[i];
  }
  delay[62]=x;
}

int main(void)
/* Hauptprogramm, Daten von Codec holen und direkt wieder ausgeben.
*/
{ PCTL.I=0x261009; // PLL des DSP setzen
  PBC = 0; // Port B als Ein/Ausgaenge konfigurieren
  PBDDR = 0x0100; // PB0..PB7,PB9..PB14 als Eingang definieren. PB8 als Ausgang.
  PBD=0x0; // PB8 auf 0. PB8 dient als GND fuer Bruecke zu PB14(HACK)
  // Codec Initialisieren: XX Sample Rate, Stereo, mit HP Filter, 16 Bit Datenwerte...
  codecInit(NO_PREAMP ! HI_PASS_FILT ! SAMP_RATE_16 ! STEREO+DATA_16 , // CTRL_WD_12
    IMMED_3STATE ! XTAL2_SELECT ! BITS_64 ! CODEC_MASTER, // CTRL_WD_34
    0x000000, // CTRL_WD_56
    0x000000 // CTRL_WD_78
  );

  while(1)
  { while( SSISR.B.IFS); //warten auf frame sync Transition
    while(!SSISR.B.IFS)
    { x = rxData.d.i.audio_left;
      if (PBD & 0x4000) // PB14-PB8 keine Bruecke?
        y = x; // Audio direkt passieren lassen
      else
        filterFIR(); // Audio filtern

      txData.d.i.audio_left=y;
      txData.d.i.audio_right=(fract)0.0; // Rechter Kanal nichts
      txData.d.i.output_setting=TONE_OUTPUT;
      txData.d.i.input_setting=TONE_INPUT;
    }
  }
  return 0;
}

File: S:\E97\DSP\DSPuebungen\Uebung3_FIR_Hochpass\FIR_HP1kHz.c* Mod Ins Line: 99 Col: 63
```