BEUTH HOCHSCHULE
FÜR TECHNIK
BERLIN
University of Applied Sciences

Realisierung und Analyse des IIR-Filters

Laborbericht

angefertigt von

Robby Kozok, Nic Frank Siebenborn, Pascal Kahlert

in dem Fachbereich VII – Elektrotechnik - Mechatronik - Optometrie –
für das Modul Digitale Signalverarbeitung III
der Beuth Hochschule für Technik Berlin im Studiengang
Elektrotechnik - Schwerpunkt Elektronische Systeme

Datum 12. Januar 2016

Lehrkraft
Prof. Dr.-Ing Marcus Purat Beuth Hochschule für Technik

Inhaltsverzeichnis

1 Fast Fourier Transformation ohne Fensterung **2**
1.1 Aufgabenstellung 2
1.2 Durchführung 2
1.3 Auswertung 5

2 Fast Fourier Transformation mit Fensterung **10**
2.1 Aufgabenstellung 10
2.2 Durchführung 10
2.3 Auswertung 10

3 DTMF **11**
3.1 Aufgabenstellung 11
3.2 Durchführung 11
3.3 Auswertung 11

Kapitel 1

Fast Fourier Transformation ohne Fensterung

1.1 Aufgabenstellung

In dieser Aufgabe soll die bereits vorhandene Implementierung der FFT untersucht werden. Dafür sollte in der Vorbereitung eine Funktion entwickelt werden, welche das Betragsspektrum ermittelt.

1.2 Durchführung

Zur Überprüfung der FFT sollten zwei unterschiedliche
Dual-tone multi-frequency(Mehrfrequenzwahlverfahren) (DTMF) Frequenzen gewählt werden und jeweils als Sinus-Signal mit einer Amplitude von 100mV auf den Eingang des DSP gegeben werden. Dazu muss der vorhandene Source-Code der ISR.c den Anforderungen der Vorbereitung angepasst werden. Die Anforderungen werden durch die Abbildung 1.1 dargestellt.

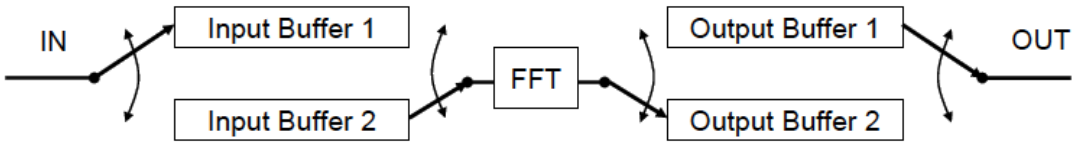

Abbildung 1.1: Segmentverarbeitung mit Wechselbuffer, Quelle: Purat, S. 17

Dabei sollte auf beiden Seiten ein zweiteiliger Buffer implementiert werden, dies hat den Hintergrund, dass die FFT immer über eine bestimmt Anzahl von Werten angewendet wird. Da die FFT aber Länger als $\frac{1}{Abtast} = \frac{1}{48kHz}$ dauert und der Buffer nicht geändert werden darf solange die FFT bearbeitet wird muss hier mit einem Arbeitsbuffer und einem temporären Buffer gearbeitet werden. Dies wurde in dem Source-Code 1.2 implementiert.

```
1 #include <fract.h>
2 #include <ccblkfn.h>
3 #include <sys\exception.h>
4 #include <cdefBF561.h>
5 #include "codeclib.h"
6 #include "isr.h"
7
8 /* Add your own global variables and definitions here */
```

```
9
10 #define TRIGGER_ON      0x7FFFFFFF
11 #define TRIGGER_OFF     0x00000000
12
13 #define INTERNAL_ADC_R1 0x5
14
15 #define INTERNAL_DAC_L0 0x0
16 #define INTERNAL_DAC_R0 0x4
17
18 char cNewFrame = 0;
19 // Pointer für die Buffer
20 fract16 *pInFrame, *pOutFrame, *pInBuffer, *pOutBuffer;
21 // Buffer Arrays
22 fract16 inBuffer1[FRAMELENGTH], inBuffer2[FRAMELENGTH],
23                 outBuffer1[FRAMELENGTH], outBuffer2[FRAMELENGTH];
24
25
26 EX_INTERRUPT_HANDLER(Sport0_RX_ISR)
27 {
28 /* Add your own local variables here */
29
30     static unsigned int iSampleCounter = 0;
31     // confirm interrupt handling
32     *pDMA2_0_IRQ_STATUS = 0x0001;
33
34 /* Add input and output and pointer setting here */
35         // Bei neuem Sample Frame müssen die Pointer umgeschaltet werden
36     if(iSampleCounter == 0){
37         if(pInFrame == inBuffer1){
38             pInBuffer   = inBuffer1;
39             pInFrame    = inBuffer2;
40             pOutBuffer  = outBuffer1;
41             pOutFrame   = outBuffer2;
42         }
43         else{
44             pInBuffer   = inBuffer2;
45             pInFrame    = inBuffer1;
46             pOutBuffer  = outBuffer2;
47             pOutFrame   = outBuffer1;
48
49         }
50         iDMATxBuffer[INTERNAL_DAC_L0] = TRIGGER_OFF;//Trigger on DACL1
51     }
52     // Schreibe den aktuellen Eingangswert in den aktuellen Eingangsbuffer
53     pInBuffer[iSampleCounter] = iDMARxBuffer[INTERNAL_ADC_R1] >> 16;
54     // Schreibe den aktuellen Ausgangswert in den aktuellen Ausgangsbuffer
55     iDMATxBuffer[INTERNAL_DAC_R0] = pOutBuffer[iSampleCounter] << 16;
56     // Zähle Samples hoch
57     iSampleCounter++;
58     // Wenn genug Samples vorhanden sind muss eine neue FFT angestoßen werden.
59     if(iSampleCounter >= FRAMELENGTH){
60         iSampleCounter = 0;
61         iDMATxBuffer[INTERNAL_DAC_L0] = TRIGGER_ON; //Trigger on DACL1
62         cNewFrame = 1;
63     }
64 }
```

Routine zum aufnehmen der Samples und umschalten der Buffer

Zu erst wurden in Zeile 20 - 22 benötigten Pointer und Arrays zum umschalten deklariert. In Zeile 36 - 49 werden je nach dem zuletzt verwendetet Buffer die Bufferadressen geändert, dies geschieht immer bei der Aufnahme des ersten Samples. Zeile 53 und 55 dienen zum Einlesen des Signals und zum Ausgeben des Spektrums. Sobald genug Samples

aufgenommen wurden wird in Zeile 59 - 62 ein Trigger gesetzt und eine Flag für die FFT gesetzt.
Im weiteren musste eine Routine zur Berechnung des Betragsspektrums erstellt werden. Diese sollte in Assembler geschrieben werden. Die Umsetzung ist in dem Source-Code 1.2 zu sehen.

```
1  .section/DOUBLE32 program;
2  .align 2;
3  _winmul:
4          // R0 = Pointer to pInFrame (fract16)
5          // R1 = Pointer to window (fract16)
6          // R2 = FFT Order
7  ._winmul.end:
8  .global _winmul;
9
10 .align 2;
11 _abs2_spec:
12         // R0 = Pointer to abs2_spectrum (fract16)
13         // R1 = Pointer to spectrum (complex fract16)
14         // R2 = FFT Order
15
16         P2 = R2; // FFT Ordnung
17         I0 = R0; // Betragsspektrum - Return
18         I1 = R1; // Komplexes Spektrum - Parameter
19
20         R3.L = 10; // Schiebe Faktor
21
22         NOP;NOP;NOP;NOP;
23
24         R1.L = W[I1++];
25
26         LSETUP(_LOOP_START, _LOOP_END) LC0 = P2; // Schleifendurchlaufanzahl
27         _LOOP_START:
28             A0      = R1.L * R1.L || R1.L = W[I1++];
29             R2.L    = (A0 += R1.L * R1.L) || R1.L = W[I1++];
30             R4.L    = ASHIFT R2.L BY R3.L;
31         _LOOP_END:
32             W[I0++] = R4.L;
33         RTS;
34 ._abs2_spec.end:
35 .global _abs2_spec;
```

Routine zum berechnen des Betragsspektrums

Im ersten Teil, von Zeile 16 - 18 mussten die Eingangsparameter übernommen werden. Diese waren zum einen die Pointer vom Komplexen Spektrum und zum Betragsspektrum, welches nun berechnet werden sollte und die Ordnung der FFT. Zur Skalierung wurde in Zeile 20 ein Schiebe Faktor von 10 festgelegt. In Zeile 24 wurde der erste Wert zu Berechnung eingelesen. Zeile 26 ist der beginn der Hardware Schleife diese soll der Ordnung entsprechend durchlaufen werden. Zeile 28 - 29 bilden die Betragsfunktion der komplexen Rechnung ab, diese ist in der Gleichung 1.1 zu sehen.

$$| z | = \sqrt{a^2 + b^2} \tag{1.1}$$

In Zeile 28 wird das Quadrat eines Wertes gebildet gebildet und der nächste Wert eingelesen. Zeile 29 realisiert die Bildung des zweiten quadrates, sowie die Addition beider Werte und das Auslesen des nächsten Werten. Die Skalierung in Zeile 30 wurde bereits in Zeile 20 vorbereitet. Am Ende der Loop wird der berechnete Wert ausgegeben und der Pointer um eine Stelle erhöht.

1.3 Auswertung

Leck-Effekt

Bevor zu Analyse der FFT kommen wollen wir noch über den Leck-Effekt reden, welcher uns im Laufe der Bearbeitung der Aufgaben aufgefallen ist. In Abbildung 1.2 ist zu sehen wie der schwarze Graph eine SI-Funktion darstellt, dies entsteht durch das Tiefpass-Verhalten des Ausganges. Unser Spektrum sollte im Idealfall genau ein Impuls sein, dadurch ist das Ausgangssignal die Impulsantwort des Ausganges, welches eine SI-Funktion ist.

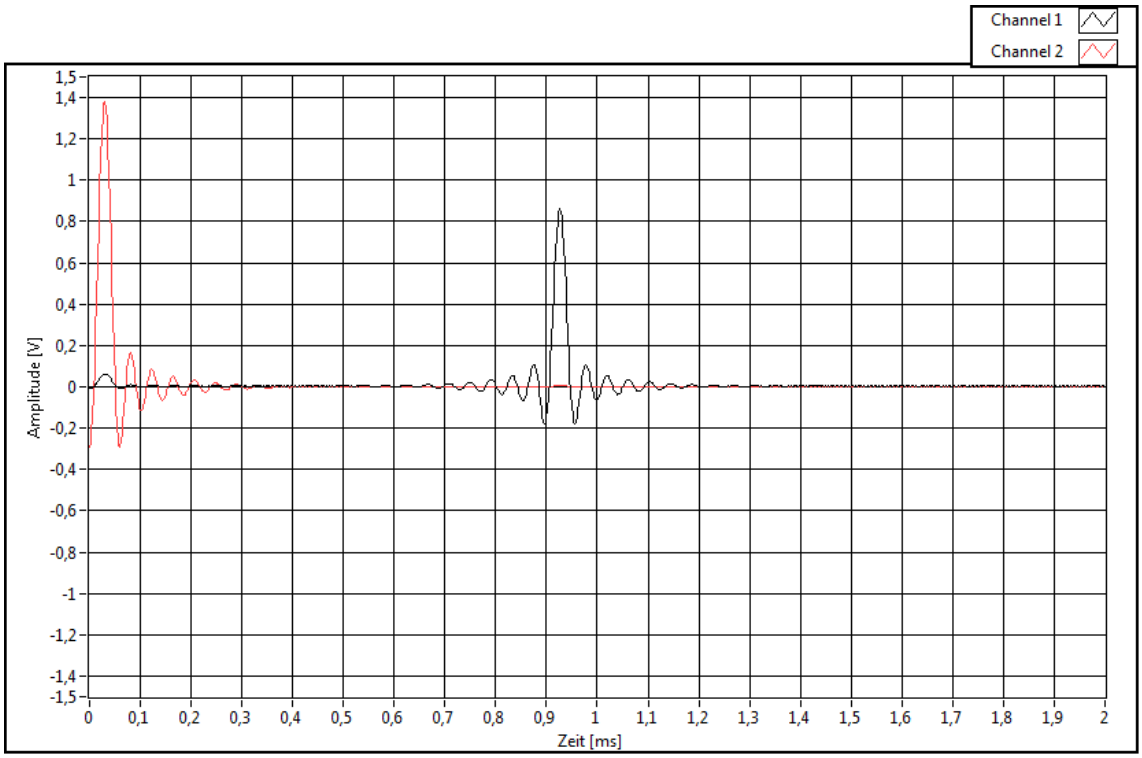

Abbildung 1.2: Ausgangssignal(505 Hz) ohne Leck-Effekt

In Abbildung 1.3 ist zu sehen, dass das Ausgangssignal keine SI-Funktion darstellt.

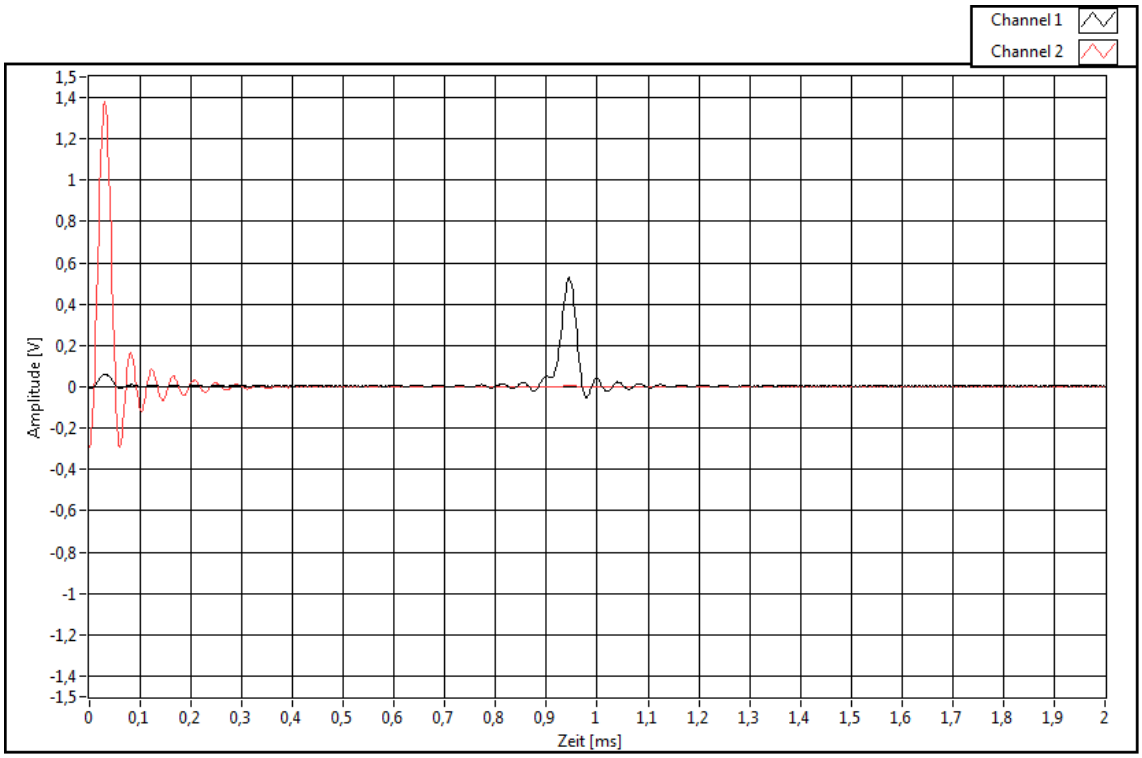

Abbildung 1.3: Ausgangssignal(511 Hz) mit Leck-Effekt

Der Leck-Effekt tritt bei Spektralanalysen auf, welche keinen unendlichen langen Beobachtungszeitraum besitzen, also praktisch bei allen. Durch den Leck-Effekt werden Frequenzen um die eigentliche Frequenz berechnet, welche nicht vorhanden sind. Dieser Effekt kann verhindert werden, indem eine Abtastfrequenz gewählt wird, welche dem ganzzahligen Vielfachen der Signalfrequenz entspricht. Natürlich sind hierbei Ungenauigkeiten zu beachten weshalb die von uns gewählten Frequenzen nicht berechnet wurden, sondern durch ausprobieren ermittelt wurden.

Mit diesem Wissen können wir nun die Analyse der DTMF Frequenzen beginnen. Als Frequenzen haben wir die ersten beiden Zeilenfrequenzen gewählt, diese sind 697 Hz und 770 Hz. Zur besseren Darstellung, haben wir jeweils ein Bild mit passender Achseneinteilung ein Detailbild genommen. Die Achseneinteilung wurde so gewählt das ein Kästchen 1,2 kHz entspricht, somit ist es möglich zu bewerten ob unsere FFT ordentlich arbeitet. In Abbildung 1.4 ist ein Impuls leicht über der Mitte des ersten Kästchens zu sehen. Dies bedeutet, dass wir davon ausgehen können das es sich hierbei um die 697 Hz handeln kann.

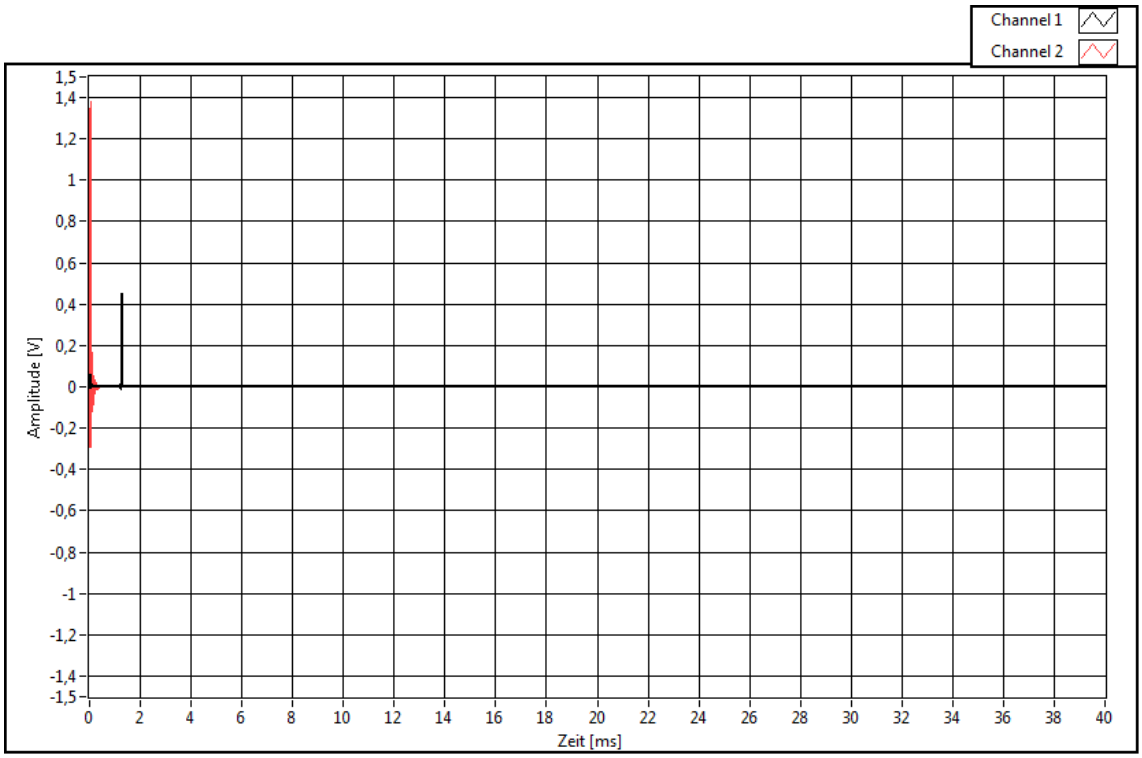

Abbildung 1.4: Ausgangsspektrum bei einem 697 Hz Eingangssignal

Abbildung 1.5 zeigt eine Nahaufnahme des Spektrums, dort ist der Leck-Effekt deutlich zu sehen. Außerdem ist nun zu sehen das sich der Impuls sehr nahe an den 697 Hz bewegt, denn $\frac{1,26ms}{2ms} * 1,2kHz = 0,762kHz$. Dies liegt augenscheinlich erstmal näher bei 770 Hz allerdings werden wir gleich sehen, dass auch das Spektrum des zweiten Signals leicht nach hinten verschoben ist.

Pascal: ung

Verschiebung erklären.

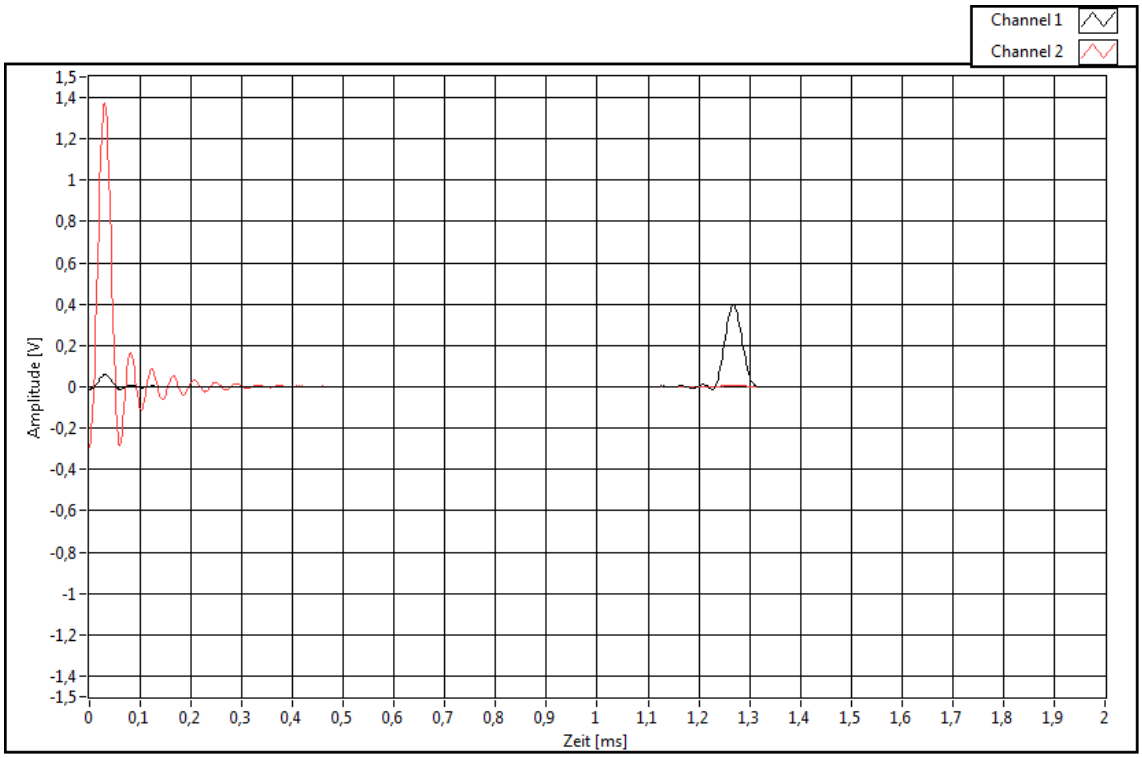

Abbildung 1.5: Ausgangsspektrum bei einem 697 Hz Eingangssignal

Abbildung 1.6 zeigt ein ähnliches Bild wie Abbildung 1.4 und weißt keine Besonderheiten auf.

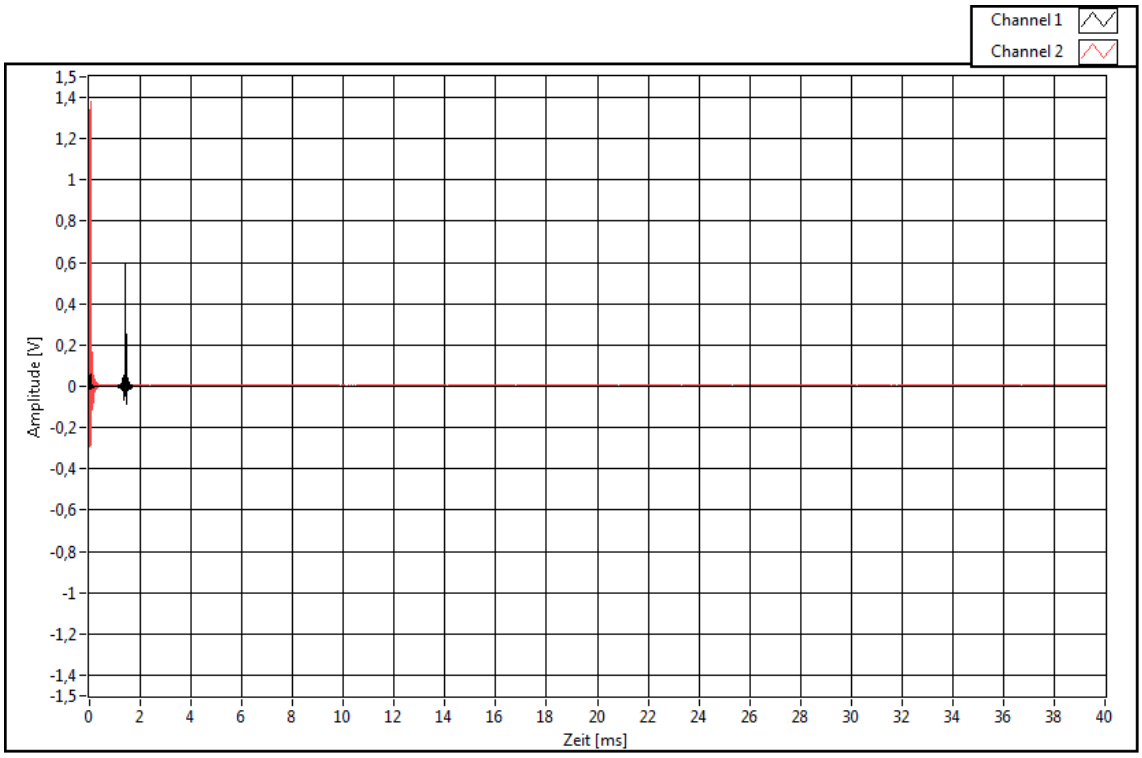

Abbildung 1.6: Ausgangsspektrum bei einem 770 Hz Eingangssignal

In Abbildung 1.7 ist zu sehen das bei 770 Hz kein oder nur ein sehr schwacher Leck-

Effekt auftritt. Außerdem ist zu sehen, dass dieses Spektrum ebenfalls weiter als erwartet nach rechts verschoben ist. An dieser Stelle wäre das Spektrum eines 840 Hz Signals zu erwarten, $\frac{1,4ms}{2ms} * 1,2kHz = 0,84kHz$

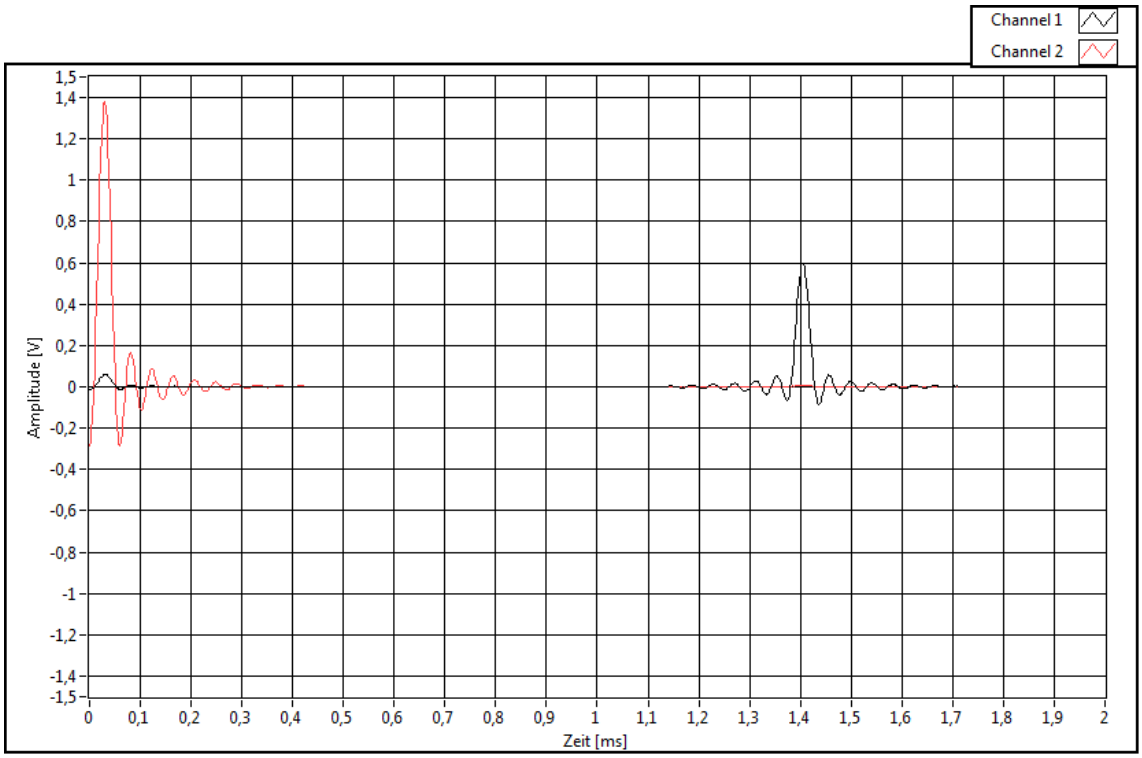

Abbildung 1.7: Ausgangsspektrum bei einem 770 Hz Eingangssignal

Kapitel 2

Fast Fourier Transformation mit Fensterung

2.1 Aufgabenstellung

2.2 Durchführung

2.3 Auswertung

Kapitel 3

DTMF

3.1 Aufgabenstellung

3.2 Durchführung

3.3 Auswertung

Abbildungsverzeichnis

1.1 Segmentverarbeitung mit Wechselbuffer, Quelle: Purat, S. 17 2
1.2 Ausgangssignal(505 Hz) ohne Leck-Effekt 5
1.3 Ausgangssignal(511 Hz) mit Leck-Effekt 6
1.4 Ausgangsspektrum bei einem 697 Hz Eingangssignal 7
1.5 Ausgangsspektrum bei einem 697 Hz Eingangssignal 8
1.6 Ausgangsspektrum bei einem 770 Hz Eingangssignal 8
1.7 Ausgangsspektrum bei einem 770 Hz Eingangssignal 9

Abkürzungsverzeichnis

DTMF Dual-tone multi-frequency(Mehrfrequenzwahlverfahren). 2