## Laboratorium zastosowania procesorów sygnałowych

# Efekty akustyczne


KATEDRA
SYSTEMÓW
MIKROELEKTRONICZNYCH

# 1. Proste operacje na sygnale audio

### Kontrola głośności.

Jedną z najprostszych operacji możliwych do wykonania w DSP na sygnale audio są zwiększanie lub zmniejszanie głośności. Dla operacji stałoprzecinkowych ta operacja może być wykonana poprzez wymnożenie próbek przez liczbę z przedziału 0x0000... 0x7FFF... lub prościej używając operacji przesunięcia bitowego (wymnożenie przez kolejne potęgi liczby 2). Gdy zwiększamy poziom głośności musimy pamiętać o niwelowaniu efektów: przepełnienie, niedopełnienie, nasycenie, efekty szumów kwantyzacji.

### Miksowanie wielu kanałów sygnałów audio.

Łączenie wielu sygnałów audio przy pomocy DSP jest łatwe. Zamiast używać obwodów sumujących opartych na wzmacniaczach operacyjnych używamy sumatora lub MAC (Multiply Accumulator). Najpierw wymnażamy sygnały przez współczynnik tak aby po ich zsumowaniu nie wystąpiło przekroczenie zakresu. Najprościej dobrać ten współczynnik licząc odwrotność liczby sygnałów, 2 sygnały wymagają współczynnika ½, a 4 sygnały ¼.
Można także miksować sygnały z różnymi wagami tych sygnałów np.

$$y(n) = \frac{1}{5}x_1(n) + \frac{1}{10}x_2(n) + \frac{3}{10}x_3(n) + \frac{1}{20}x_4(n) + \frac{9}{20}x_5(n)$$

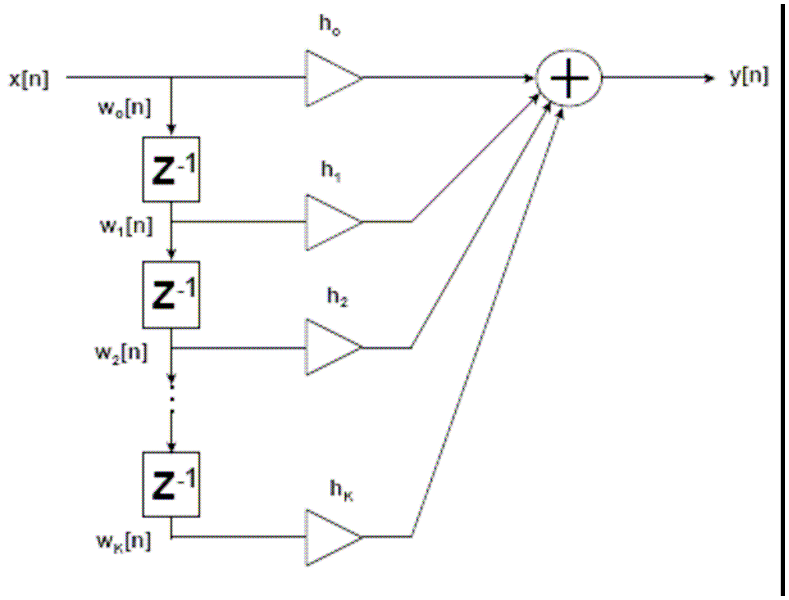
Suma wszystkich współczynników nie powinna przekroczyć 1.

# 2.Techniki filtracji.

Jednym z głównych zastosowań DSP w audio jest filtrowanie cyfrowe. Filtry cyfrowe są używane do podwyższania lub obniżania poziomu sygnału w pewnych zakresach częstotliwości, podobnie jak ma to miejsce w korektorze w stereo. Filtry te podobnie jak analogowe podzielone są na: dolnoprzepustowe, środkowoprzepustowe , środkowozaporowe, górnoprzepustowe. Mogą być one projektowane jako FIR (o skończonej odpowiedzi impulsowej) lub IIR (o nieskończonej odpowiedzi impulsowej). Używając tych dwóch podstawowych filtrów w różnych konfiguracjach możemy zbudować cyfrowe odpowiedniki analogowych filtrów takich jak korektory parametryczne, korektory graficzne, filtry grzebieniowe.

### Filtry FIR

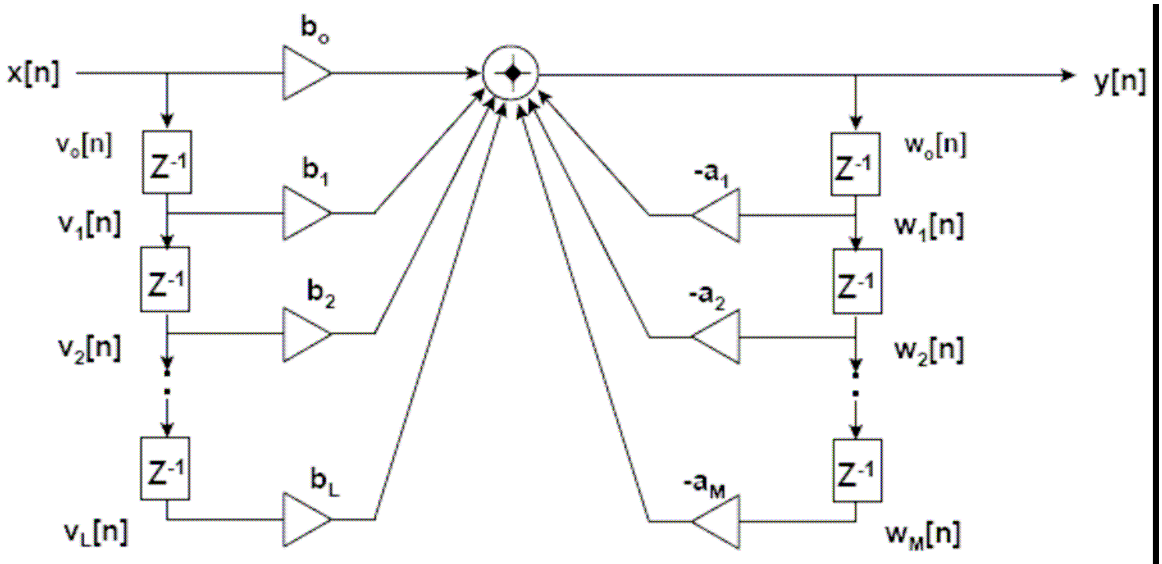
Filtr FIR ma odpowiedź impulsową skończoną w czasie zależną od stopnia filtra. Sygnał wyjściowy obliczany jest na podstawie poprzednich wartości sygnału tak jak jest to ukazane na grafie poniżej.




$$y[n] = \sum_{k=0}^{\infty} h[k]x[n-k]$$

**Filtry IIR**

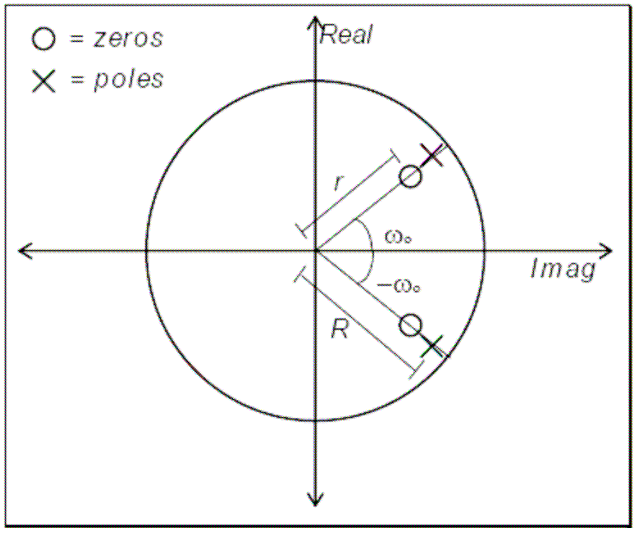
Filtry IIR mają odpowiedź impulsową, która jest nieskończona w czasie. Wartość wyjścia (y[n]) obliczana jest na podstawie poprzednich wartości sygnału wejściowego (x[n]) i wyjściowego (y[n]). IIR porównuje się do rekursywnych filtrów.

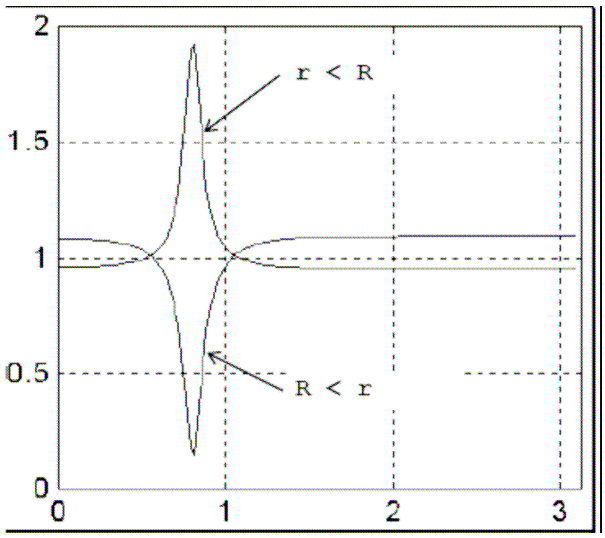



$$y[n] = \sum_{i=1}^{M} \left(-a_i y[n-i]\right) + \sum_{j=0}^{L} \left(b_i x[n-j]\right)$$

**Filtry parametryczne**

Filtry parametryczne są używane dość często w obróbce audio ze względu na to, że potrafią wzmacniać lub tłumić pewne składowe częstotliwościowe sygnału. Pierwotnie filtry tego rodzaju budowane były przy użyciu komponentów analogowych**.** Ich cyfrowa implementacja jest bardzo prosta i wydajna. Pasmo, częstotliwość środkowa i wzmocnienie może być wyznaczone przy użyciu prostych wyrażeń wymagających 4 współczynników.
Filtry parametryczne są filtrami IIR drugiego rzędu i mają odpowiedź częstotliwościową zawierającą pojedyncze maksimum lub minimum dla częstotliwości $\omega_0$. Wzmocnienie dla innych częstotliwości jest jednostkowe. Filtry te w wymagają mniej mocy obliczeniowej niż filtry FIR i IIR wyższych rzędów, a także wielkość mocy obliczeniowej wymaganej do obliczenia współczynników jest minimalna w porównaniu filtrów wyższych rzędów.
Oto jak działają filtry: sprzężone pary biegunów i sprzężone pary zer są ułożone na linii biegnącej od początku płaszczyzny jak to pokazano na rysunku. Jeśli bieguny są bliżej początku układu niż zera to wynikowy filtr ma minimum. W przeciwnym wypadku, gdy zera są bliżej początku układu niż bieguny filtr ma maksimum. Na drugim rysunku pokazane są charakterystyki częstotliwościowe obu filtrów.




Transmitancję takiego filtru można opisać:

$$H(z)=\frac{Y(z)}{X(z)}=\frac{b_0+b_1z^{-1}+b_2z^{-2}}{1-a_1z^{-1}-a_2z^{-2}}$$

Wyrażenie różnicowe:

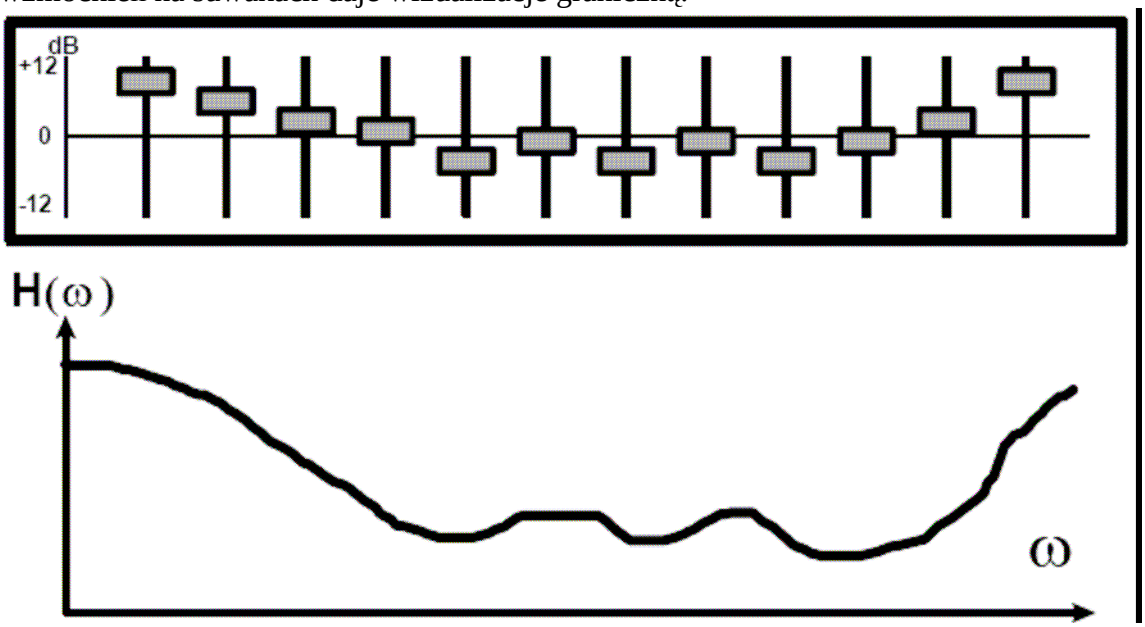
$$y(n)=b_0x(n)+b_1x(n-1)+b_2x(n-2)+a_1y(n-1)+a_2y(n-2)$$

Wyrażenie różnicowe jest zaimplementowane w przykładowym kodzie zawartym poniżej. Następujące wyrażenia (Orphandis, 1996) są użyte to obliczenia wartości współczynników na podstawie $\omega_0$, r oraz R.

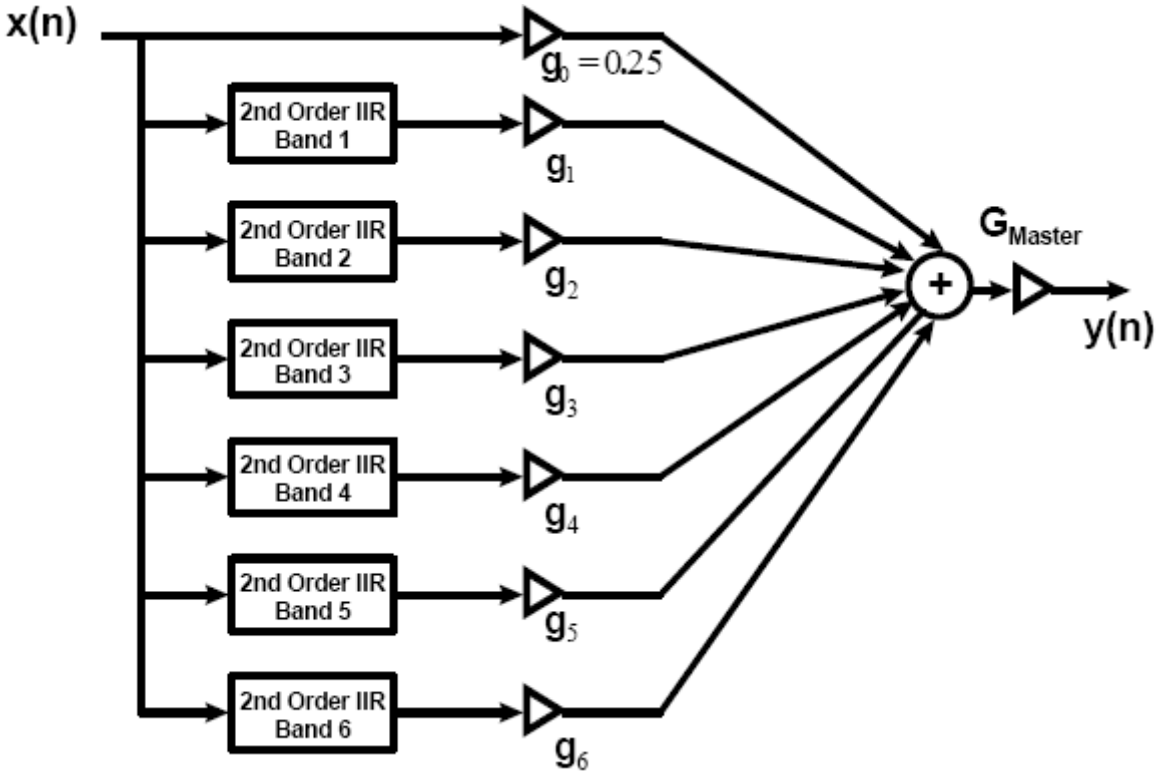
$$a_0=1 \qquad a_1=-2R\cos(\omega_0) \qquad a_2=R^2$$
$$b_0=1 \qquad b_1=-2r\cos(\omega_0) \qquad b_2=r^2$$

**Korektory graficzne**

W zastosowaniach profesjonalnych i konsumenckich korektory są używane do zmiany amplitudy określonych częstotliwości. W korektorach graficznych pasmo częstotliwościowe jest podzielone na zakresy przy użyciu pasmowoprzepustowych filtrów. Ustawienie różnych wzmocnień na suwakach daje wizualizacje graniczną.

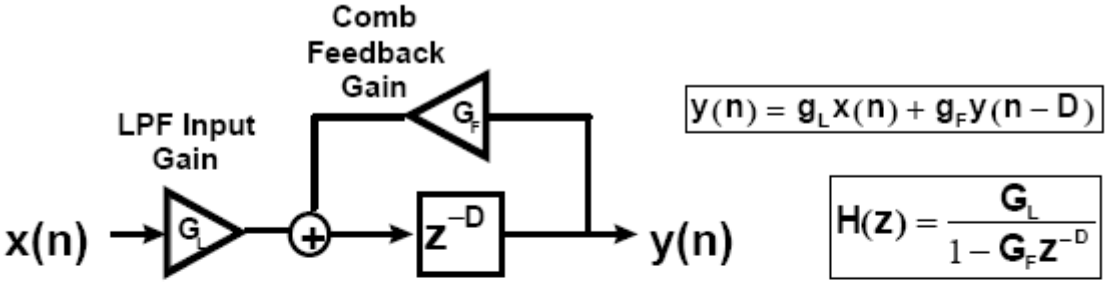


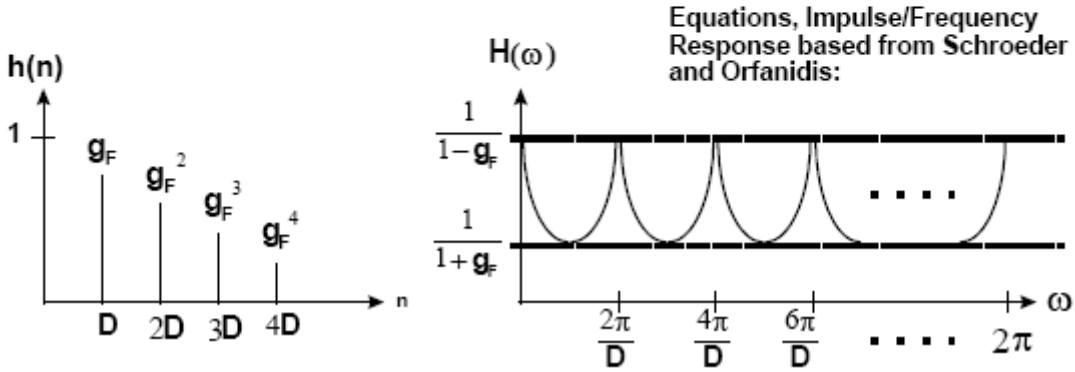

Rozwiązania analogowe wykorzystują elementy pasywne i aktywne. Zwiększenie ilości zakresów skutkuje rozrośnięciem się płytki n której wykonany jest układ. W przypadku korektora cyfrowego wymagana jest jedynie większa moc procesora, podczas gdy zajętość miejsca pozostaje taka sama.
Przykładowa struktura korektora może wyglądać jak na rysunku poniżej:



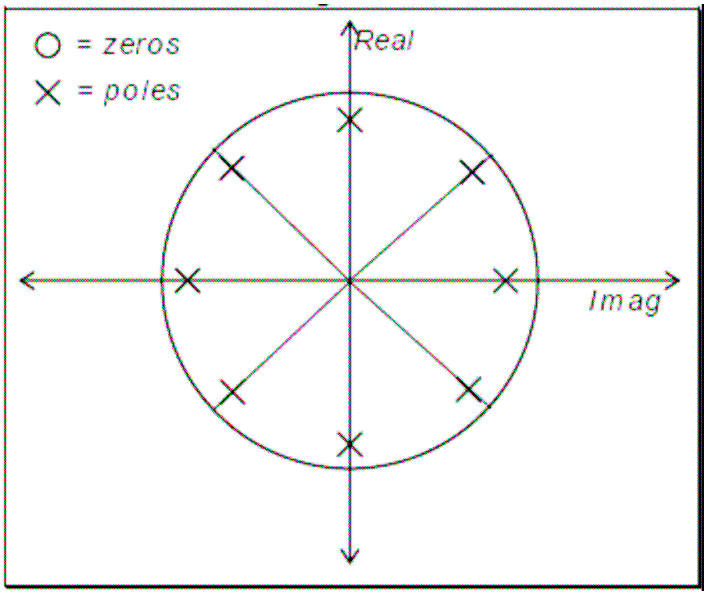

**Filtry grzebieniowe**

Filtry grzebieniowe są używane do redukcji szumów sygnałów okresowych, poprawiania jakości sygnałów, uśredniania sygnałów i są wymagane przy używaniu efektów dźwiękowych takich jak opóźnienia, chorus i flanger. Filtry grzebieniowe tworzone są poprzez sumowanie sygnału z jego opóźnioną I przeskalowaną wersją. Efektem tego jest to, że niektóre częstotliwości są tłumione a niektóre wzmacniane. Filtry grzebieniowe symulują wielokrotne odbicia sygnałów dźwiękowych. Jak sama nazwa mówi, odpowiedź częstotliwościowa wygląda jak zęby grzebienia.
Oto przykład filtra grzebieniowego opartego na IIR:



Poniżej znajduje się wykres zero-biegunowy odpowiadający powyższej strukturze. Odległość od biegunów do początku układu jest określona przez $G_F$. Liczba biegunów jest powiązana z długością elementów opóźniających albo z wartością D. W przykładzie poniżej D=8, $G_F \approx 0.9$.




Orfandis opracował filtry grzebieniowe i ich przykładowe użycie. Przedstawił on bardzo dobry przykład.

$$\mathbf{H_{IIRcomb}(z) = b\frac{1+z^{-D}}{1-az^{-D}}}, \qquad \mathbf{H_{FIRcomb}(z) = \frac{1}{N}(1+z^{-D}+z^{-2D}+\ldots.z^{-(N-1)D}) = \frac{1}{N}\frac{1-z^{-ND}}{1-z^{-D}}}$$

Dla FIR może być użyta do dodawania opóźnionej wersji sygnału dla efektów opóźnienia.
Dla D=1, otrzymujemy FIR wygładzający, albo przesuwający filtr uśredniający.

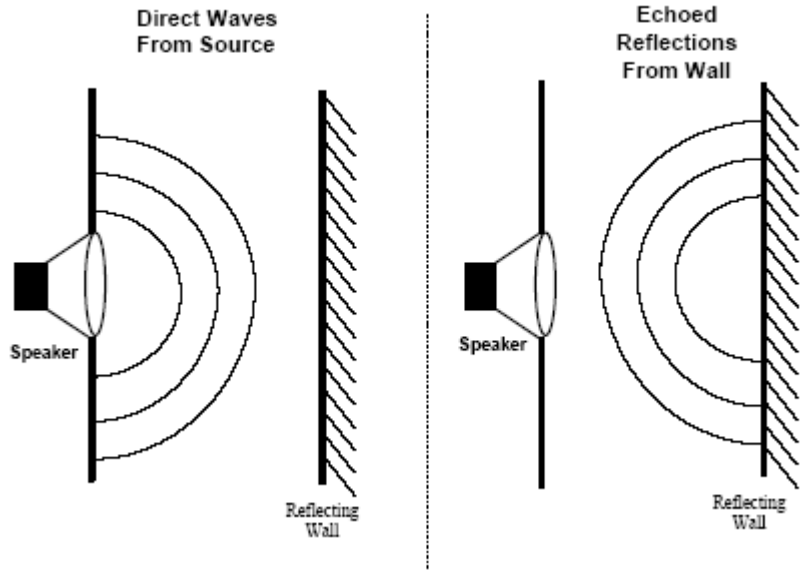
$$\mathbf{H_{FIR\_MA}(z) = \frac{1}{N}(1+z^{-1}+z^{-2}+\ldots.z^{-(N-1)}) = \frac{1}{N}\frac{1-z^{-N}}{1-z^{-1}}}$$

Rekursywne filtry grzebieniowe mogą być stosowane pryz tworzeniu echa:

$$\mathbf{H_{COMB}(z) = \frac{1}{1-az^{-D}}}$$

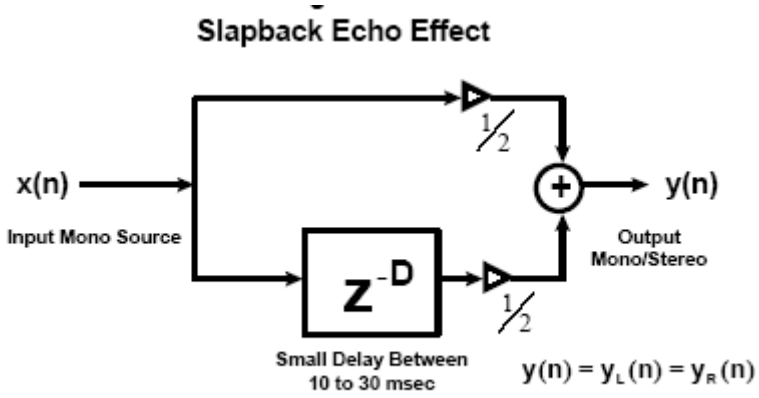
## 3. Cyfrowe opóźnienia

Cyfrowe opóźnienie jest najprostszym z opóźnień czasowych. Efekt opóźnienia jest podstawą efektów bardziej złożonych takich jak Langer czy Stereo Chorus, które zmieniają opóźnienie w przelocie. Odbicia są więc używane jako podstawa efektu wczesnego odbicia i rekursywnych opóźnień.




**Automatic Double Tracking (ADT) and Slapback Echo**

Jednym z zastosowań cyfrowego opóźnienia jest szybkie powtórzenie sygnału wejściowego z pojedynczym odbiciem. Dzięki opóźnieniu 15-40ms otrzymujemy efekty doubling slapback. Slapback pozwala na pogrubienie dźwięku np. instrumentu gdy zmieszamy go z sygnałem monofonicznym.

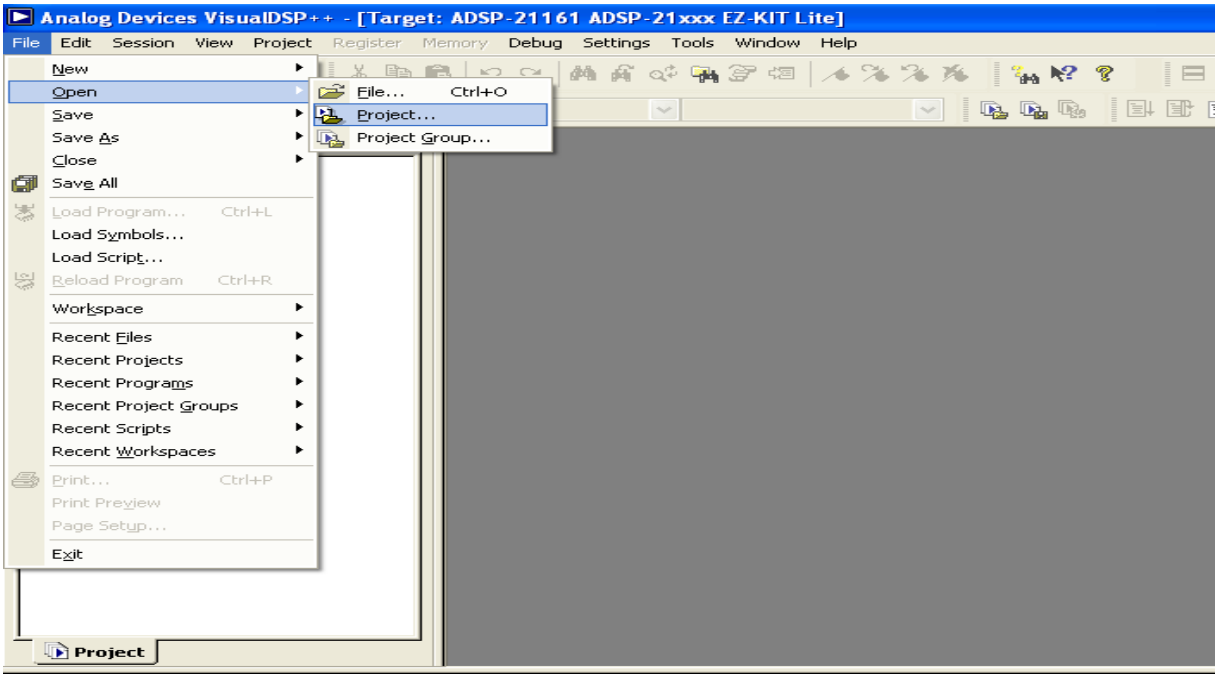


## Opis środowiska

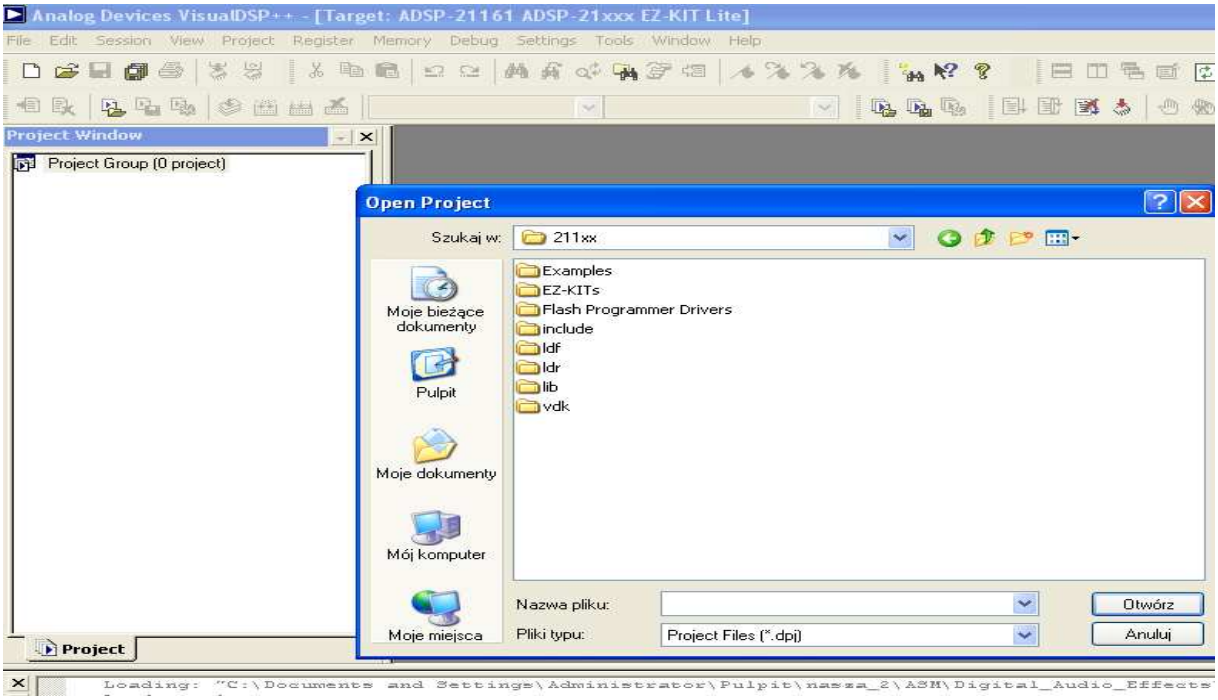
1.Środowiskiem, którego będziemy używać jest Analog Devices Visual DSP++. Aby uruchomić projekt trzeba przejść szereg kroków:

2.Włączamy program poprzez dwukrotne klikniecie ikony Analog Devices Visual DSP++ znajdującej się na pulpicie.

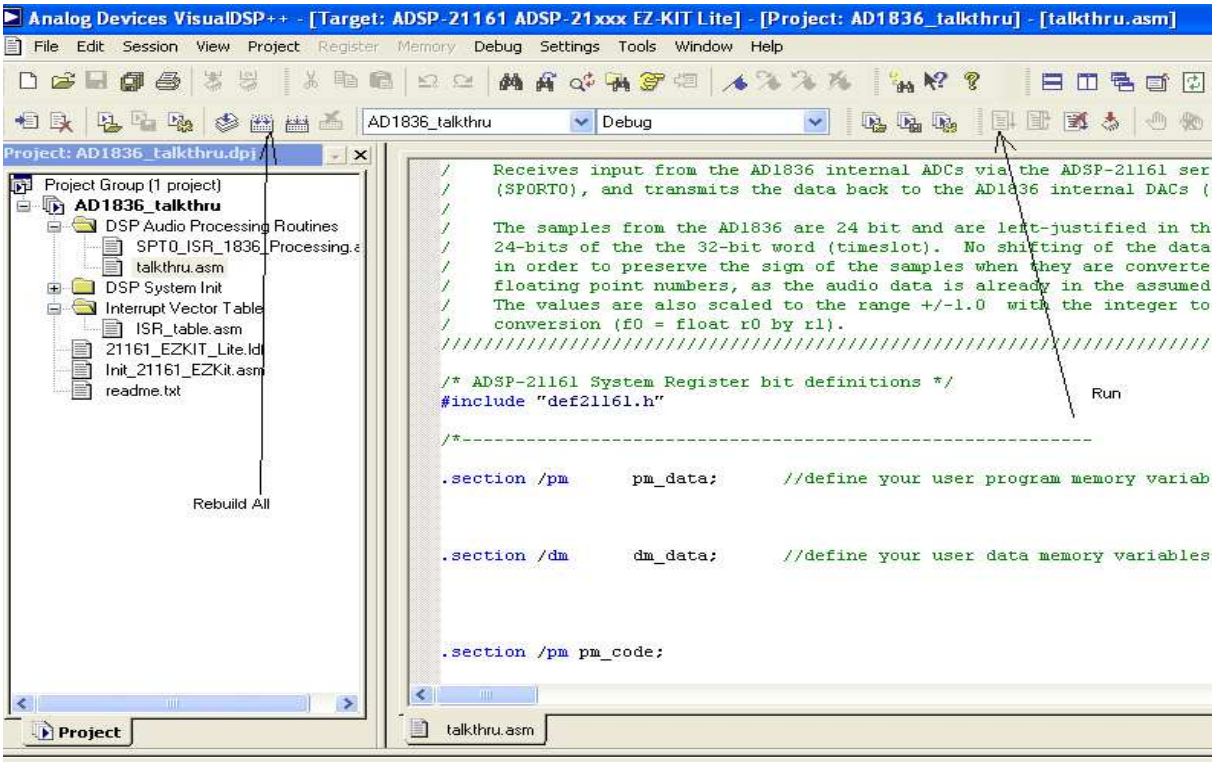
Wybieramy menu File-Open-Project i wybieramy projekt, który nas interesuje.




3.Aby tego dokonać wchodzimy do katalogu 211xx dalej EZ-KIT



4. Wybieramy interesujący nas projekt, musimy go teraz zbuildować (Build) i uruchomić (Run) możemy tego dokonać za pomocą przycisków nawigacyjnych lub wybierając odpowiednią opcję z menu. Po załadowaniu obserwujemy efekty na wyjściu zestawu.




## Przykład Talk-through

Ten przykład to szkielet pozwalający użytkownikowi na budowanie innych projektów audio. Jest to prosty sterownik dla AD1836, który bierze próbkę z bufora wejściowego i kopiuje ją do bufora wyjściowego. W czasie pracy płytka po prostu "przepuszcza" dźwięk konwertując go najpierw A/C, a potem z powrotem C/A.

Funkcja Process_Audio importuje i eksportuje próbkę dźwięku. Funkcja odbioru wyrównaną do lewej (left-justified) stało przecinkową próbkę wejściową i konwertuje ją do formatu zmiennoprzecinkowego.
W funkcji wyjściowej dzieje się odwrotnie.

W tym projekcie, pomiędzy funkcje odbioru i nadawania można wstawić odwołanie do innego algorytmu do przetwarzania danych (np. filtr lub opóźnienie cyfrowe).
Wiele z programów dostarczonych z tym hardwarem wykorzystuje ten projekt jako swoją podstawę.

**Zadania do wykonania**

- Podłącz głośniki do Ch 2 (Line Out)
- Podłącz źródło dźwięku (mic lub line)
- W menu "Project" wybierz "Build Project".
- Otwórz sesję ADSP21161 EZ-KIT Lite w debuggerze VisualDSP++.
- Załaduj "AD1836_TALKTHRU.DXE"
- Uruchom projekt i zacznij mówić do mikrofonu by usłyszeć efekt.

- Opóźnij sygnał poprzez odpowiednie zmodyfikowanie kodu:

```
void	Process_Samples( int sig_int)
{
	Receive_Samples();

//	Left_Channel_Out0 = Left_Channel_In1;
//	Right_Channel_Out0 = Right_Channel_In1;

	/* create a simple stereo digital delay on internal AD1836 stereo DAC1 channel */
	Right_Channel_Out0 = DelayLine[Index] + Right_Channel_In1; // delayed left + right  channel
```

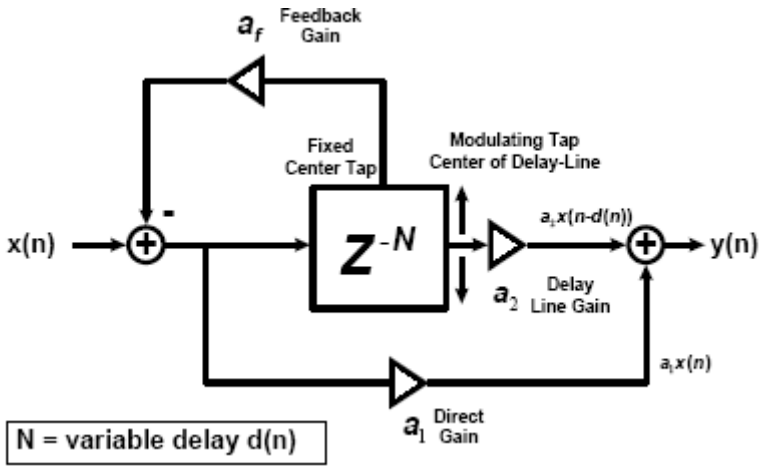
```
//Dźwięk odpowiednio na prawym lub lewym kanale:
void	Process_Samples( int sig_int)
{
	Receive_Samples();
	Left_Channel_Out0 = Left_Channel_In1;		//poprzez wyremowanie tej linii tylko prawy kanał
	Right_Channel_Out0 = Right_Channel_In1;		//poprzez wyremowanie tej linii tylko lewy kanał

	Right_Channel_Out1 = DelayLine[Index] + Right_Channel_In1;
	Left_Channel_Out1 = Left_Channel_In1;
	DelayLine[Index++] = Left_Channel_In1;
	if (Index == 12000) Index = 0;
	/* loop back other audio data */
	Left_Channel_Out2 = Left_Channel_In0;
	Right_Channel_Out2 = Right_Channel_In0;
	Left_Channel_AD1852 = Left_Channel_SPDIF_rx;
	Right_Channel_AD1852 = Right_Channel_SPDIF_rx;
```

## 4. Efekty modulacji

Opóźnieniowe efekty modulacji są czasem bardzo interesującymi efektami audio, ale nie są złożone obliczeniowo. Technika używana jest często jako interpolacja linii opóźniającej, gdzie linia opóźniająca jest przeważnie modyfikowana przez niskie częstotliwości. Rezultatem interpolacji w granicach próbki linii opóźniającej lekko zmieniamy wysokość dźwięku na wejściu sygnału. Wysokość dźwięku w algorytmie może przesuwać do innej kategorii, mimo że są inne metody DSP do tego celu. Poniżej jest lista niektórych efektów opóźniających linie modulacyjna.

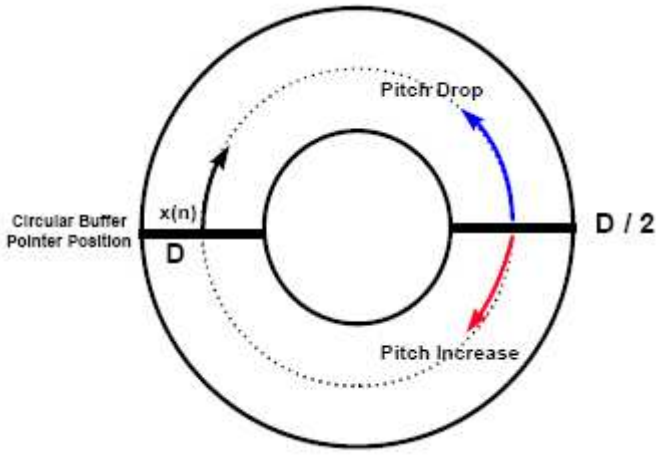
Stereo chorus, Flanger, Detune, Doubling, Leslie Rotating Speaker Emulation.

Efekty modulacji zezwalają na tworzenie wielu odmiennych typów efektów, opóźniających modulacje. Każdy wejściowy dźwięk jest przechowywany w linii, opóźniającej podczas ruchu wyjściowego, ( odmienne położenie w buforze obrotowym, kiedy wahania małych opóźnień są miksowane ( mieszane) z kierowanym dźwiękiem.




rys 50

Rysunek .50 Ogólna struktura linii opóźniającej modulacje




rys 51

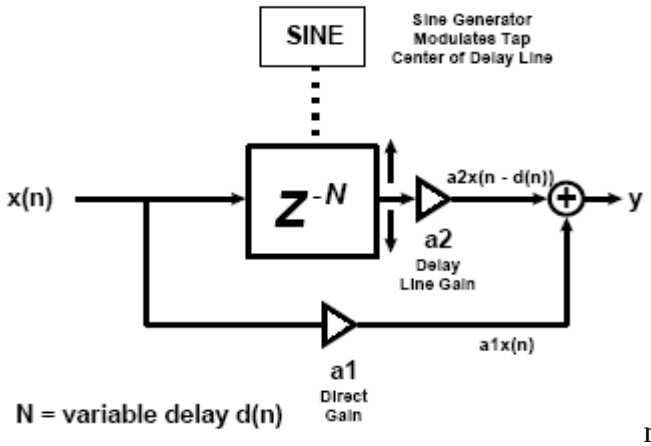
Rysunek .51 Wizualizacja wyniku centralnego tap (bufor obrotowy)

Jak widać powyższa struktura pozwala na stworzenie wielu różnych typów efektów opróżnienia modulacji. Każda wejściowa próbka jest przechowywana w linii opóźniającej podczas poruszania wyjścia otrzymany zostanie z różnych lokacji w buforze okrężnym z centralnego TAP. Jeżeli opróżnienie sygnału wejściowego jest bardzo małe ok. 10 ms. Echo pomieszane z bezpośrednim dźwiękiem spowoduje, że pewne częstotliwości zostaną pominięte w związku z filtrowaniem COMB. To spowoduje zmiany częstotliwość odpowiedzi poprzez zmiany wysokości opóźnienia podczas miksowania i opóźniania sygnału razem. Zmiany opóźnienia linii tworzą kilka niesamowitych efektów dźwiękowych.

**Flanger Effect**

Efekt ten został przypadkowo odkryty według legendy inżynier nagrywał na dwóch taśmach magnetofonowych dźwięk i monitorował odtwarzanie dwóch kaset w tym samym czasie. Podczas symulacji ADT i dublowaniu efektu został odkryte małe zmiany na taśmach. Różne szybkości odtwarzania dźwięku stworzyły „swooshing". Efekt został polepszony przez spowalnianie odtwarzania z jednej taśmy. Bardzo łatwo odtworzyć ten efekt używając DSP Flanger może być odtworzony w DSP przez różny sygnał wejściowy z małą zmianą czasu opóźniania z bardzo niską częstotliwością między 0.25 a a25ms.Przez dodanie repliki oryginalnego sygnału wejściowego rys.52., Gdy czasy opóźnienia (offset) są różne przez zmianie opóźnienia centralnego TAPA, częstotliwości w fazie i nie w fazie jako rezultat filtrowania COMP, powodują podnoszenie opadanie widma częstotliwości( efekt Swooshing – odrzutowego silnika)

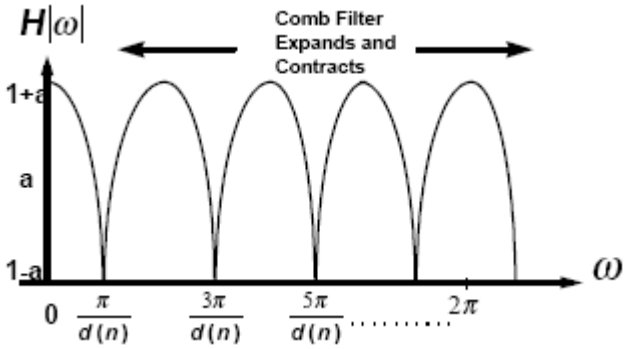
**Interpretacja efektu Flanger**




rys 52

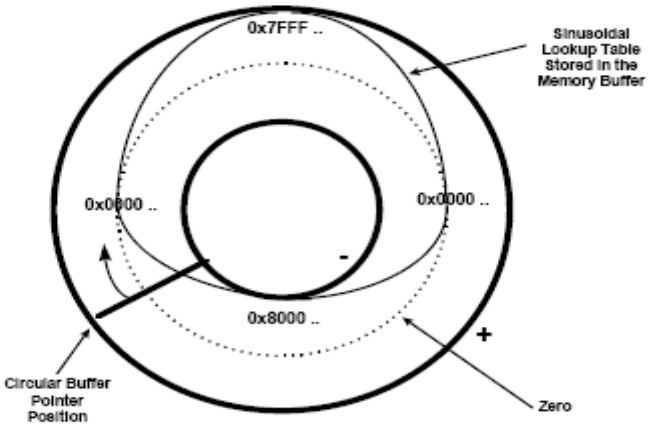
Poprzez modyfikowanie pojedynczego echa równania
y(n)= x(n)+ax( n - d(n))
Skalowanie każdego sygnału równo przez 0.5 zapobiega przesyceniu
y(n) = 0.5 [x(n) + x(n - d(n))]
Flanger jest tworzony przez periodyczne zmieniające się opóźnienia d(n). Różnice czasu opóźnienia ( lub zmiany bufora opóźniającego) mogą być łatwo kontrolowane w, DSP który używając nisko częstotliwościowego oscylatora fali sinus ( rys 54 i 55) która oblicza różnice na podstawie przykładowych próbek lub za pomocą tajmera na DSP. Aby sinusoidalnie rozróżnić opóźnienia 0< d(n) <D tajmer DSP ma system obsługi przerwań. Obsługa zdarzenia tajmera powinna obliczyć równanie
d(n) = D/2 [1 – cos (2∏f)]

Sinusoidalne LFO częstotliwości jest zazwyczaj kontrolowane przez parametr sweep rate LFO może być Łatwo zaimplementowane w DSP przez tworzenie sinusowej tablicy rys 54 która determinuje różnice w czasie opóźnienia. Determinowanie tego opóźnienia może być zmieniane periodycznie używając programowalnego timera na chipie.
Odpowiedz częstotliwościowa efektu Flanger



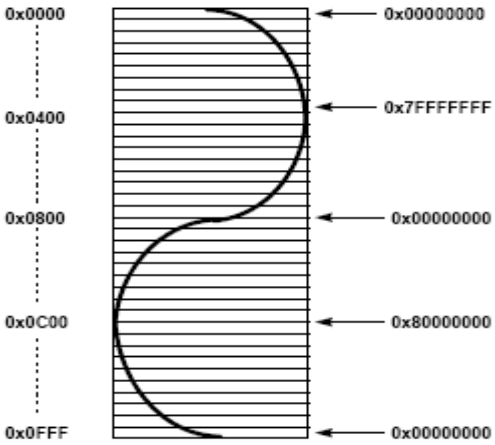

rys 53

Sinusowa tablica falowa bufora okrężnego

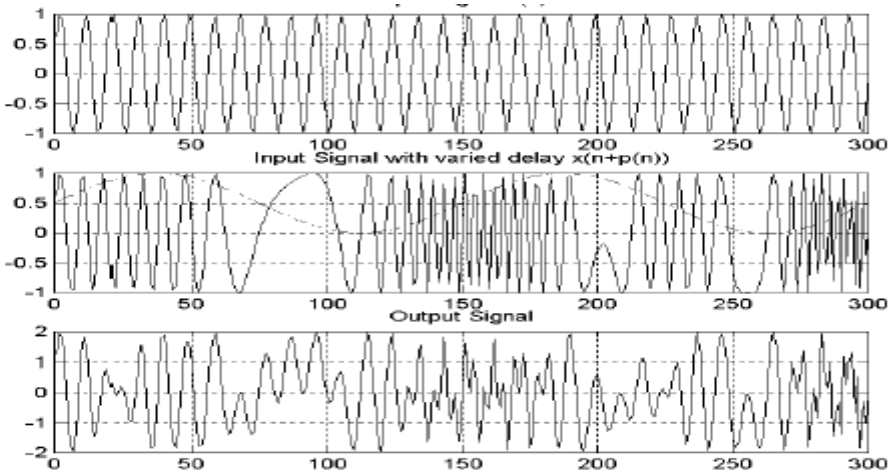



rys 54

Przykład 4k słowa w tablicy sinusoidalnej




rys 55

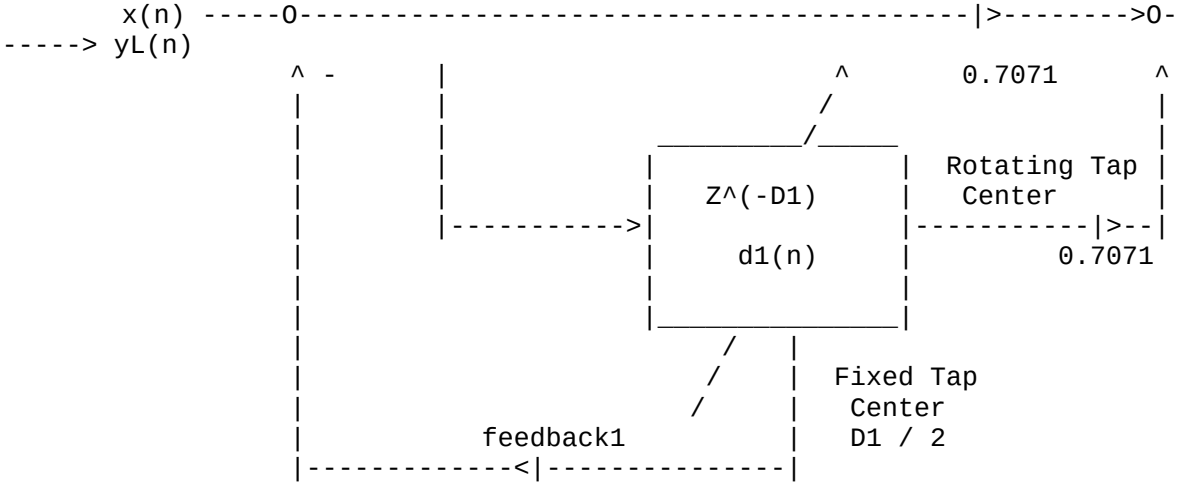


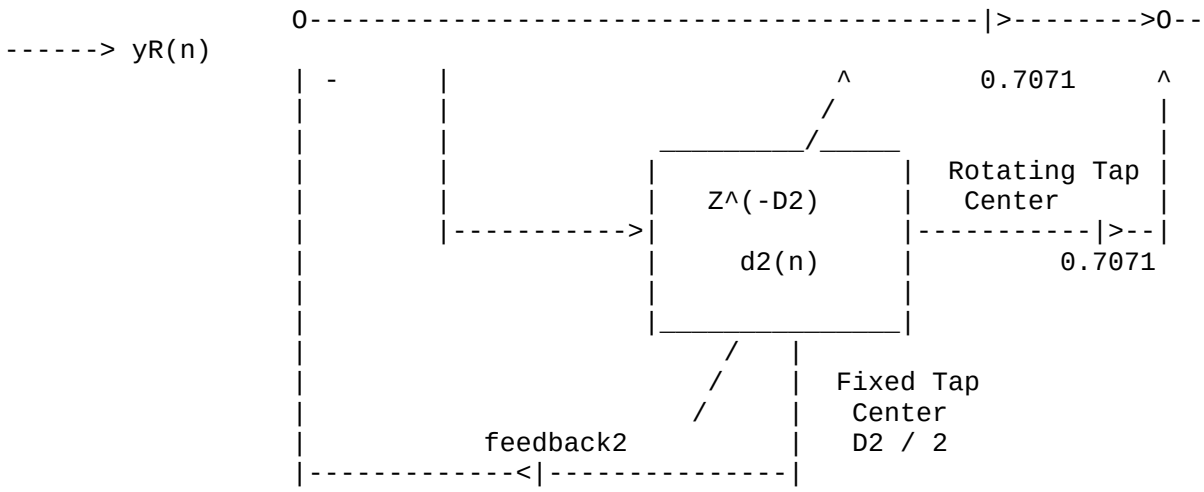

## Przykład Stereo Flanger Effect

Bazą tego projektu jest przykład "talkthrough" (przeniesienie wejścia na wyjście bez modyfikacji) dla procesora 21161. Funkcja modyfikująca jest wklejona w funkcję Talkthrough.

Równania I/O: yL(n) = 0.7071 * x(n) + 0.7071*xL(n - d1(n)) - 0.7071*xL(n - D1/2)

yR(n) = 0.7071 * x(n) + 0.7071*xR(n - d2(n)) - 0.7071*xR(n - D2/2)






Flanging to modulowanie opóźnionej o kilka milisekund kopii sygnału wejściowego i dodanie jej do sygnału oryginalnego. Spowoduje to przesunięcia fazowe i "rozjeżdżanie się" sygnału.
Opóźnienie jest modulowane za pomocą wolnozmiennej sinusoidy. Efekt ten najlepiej stosować w gitarach, bębnach i u niektórych solistów

Dla każdej próbki algorytm:
zachowuje próbkę wejściową s0 w buforze linii opóźniającej flangera - *p = s0 = xinput
generuje losowe opóźnienie, d = (D - D * sin(2*pi*fc*t)) / 2
s1 = delayed sample = tap(D, w, p, d)
y = a0 * s0 + a1 * s1

**Zadania do wykonania**

- Podłącz mikrofon do Mic In i głośniki do Ch2 (Line Out) w 21161 Ezkit Lite
- Otwórz projekt "Stereo_Flanger_Effect.dpj" w VisualDSP++ Integrated Development Environment (IDE).
- W menu "Project" wybierz "Build Project".
- Otwórz sesję ADSP21161 EZ-KIT Lite w debuggerze VisualDSP++.
- Załaduj "Stereoflanger.dxe"
- Uruchom projekt i zacznij mówić do mikrofonu by usłyszeć efekt.
- IRQ 1 i IRQ 2 zmieniają ustawienia efektu.

| /* Patch Name | DRY GAIN | WET GAIN | FEEDBACK GAIN | SWEEP RATE | SWEEP WIDTH |
|---|---|---|---|---|---|
| Slow Flange | 0x7FFFFFFF | 0x7FFFFFFF | 0x00000000 | 1 | |
| Feedback Flange | 0x7FFFFFFF | 0x7FFFFFFF | 0x5A82799A | 1 | |
| Inverted Flange | 0x7FFFFFFF | 0x80000000 | 0x00000000 | 1 | |
| Invert Flange w FB | 0x7FFFFFFF | 0x80000000 | 0xA57D8666 | 1 | |
| Medium Flange | 0x7FFFFFFF | 0x7FFFFFFF | 0x10000000 | | |
| Fast Flange | 0x7FFFFFFF | 0x7FFFFFFF | 0x10000000 | 10 | |

```
.var      IRQ1_counter = 0x00000004;
.var      IRQ2_counter = 0x00000004;
.var      feedback_gainL = 0x10000000;
 var      feedback_gainR = 0x10000000;

 var      sweep_rate = 10;
```

```
.var    sweep_widthL = 0;
.var    sweep_widthR = 0;
```

-Zmieniając wartości feedback gain , sweep rate zaobserować różnice w działaniu efektu
-Zmień wartość sweep_width oraz rate sygnału i zaobserwuj działanie efektu Flanger

```
delay_settings_1:
                r14 = 1;        DM(sweep_rate) = r14;           --zmiana tempa
                r14 = 0;        DM(sweep_widthL) = r14;         --zmiana szerokości kanału
                r14 = 0;        DM(sweep_widthL) = r14;
                ustat1=DM(IOFLAG);
                bit clr ustat1 0x3E;
                bit set ustat1 0x01;
                dm(IOFLAG)=ustat1;
        jump done_ratewidth_change;
```

- Wyłączenie  kanału prawego

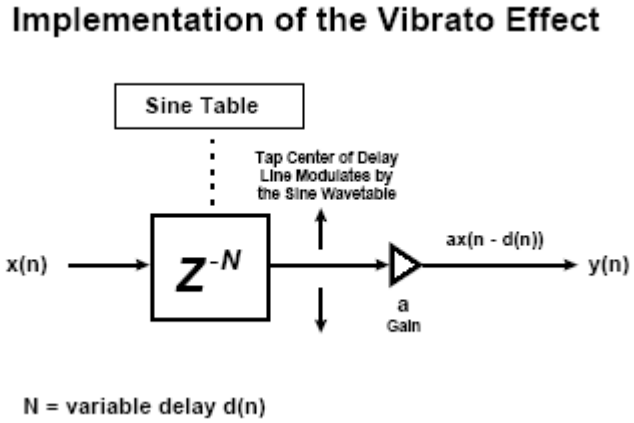
```
//      DM(Right_Channel_Out0)=r10;                 --wyłączenie prawego kanału
```

- FLAG3 pomija efekt.
- Poprzez odpowiednią modyfikacje kodu zmieniamy ustawienia Langera

**Detune Effect –** jest aktualną wersją shiftera zmieniającego wysokość dźwięku. Dźwięk jest ustawiany tak, aby różnic się od wejściowego sygnału o +/- 1%  wejściowej częstotliwości. Jest to robione przez ustawienie współczynnika shiftera: od 0.99 do 1.01. Wynikiem efektu jest zwiększenie lub zmniejszenie wejścia i łączenie zmiany wysokości dźwięku wejściowego, aby rozróżnić częstotliwość kilku Hz. Rezultatem tego jest efekt Detune. Algorytm używa mieszania efektu chóru z przebiegiem piły „Sawotooth" aby modulować opróżnienie linii. Małe aktualne wysokości dźwięku tworzą podobny efekt chorous i imitują dwa instrumenty lekko rozstrojone. Ten efekt jest przydatny w nagraniach wokalnych, aby uzyskać  wrażenie śpiewania dwóch osób używając jednego głosu. Zmiana wysokości dźwięku daje  małe zmiany wokalu (głos brzmi realnie w dalszym ciągu). Silny efekt zmian wysokości dźwięku w 5 – 10 Hz. Dla słabszych 2 -3 Hz.

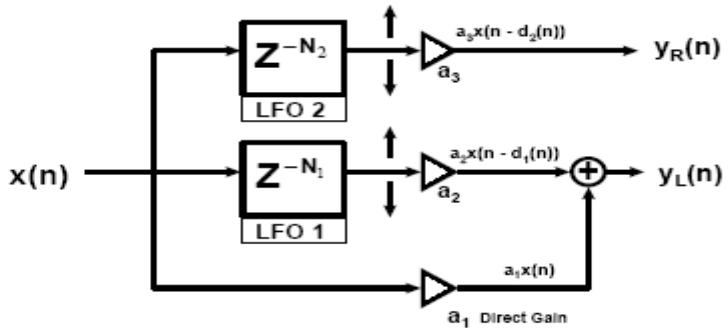
**Effect Vibrato**
Efekt vibrato podwaja drgania w glosie "wibrato" zachowując jednocześnie ton głosu. Jest to efekt podobny do zjawiska częstego dla gitarzystów gdzie muzyk używający ( chwytaka do chwytów barowych) przesuwa go po całym gryfie gitary, efekt ten jest osiągnięty przez modulowanie wysokości dźwięku. Dźwięk, który w ten sposób powstaje jest nieco zniekształcony. Delikatna zmiana wysokości dźwięku może być osiągnięty przez rozróżnianie głębokości z wystarczającą modulacją, aby stworzyć wystarczającą oscylacje. Osiągane jest to przez zmianie wysokości modulacji linii opóźniającej czasie rzeczywistym. A wybrana wartość  jest wyznaczana wg. tabeli 4k. Rezultatem tego jest interpolacja przechowywanych próbek poprzez obracanie środka TAPA linii opróżniającej. Zachowana historia próbek odtwarzana jest wstecz w wolniejszym albo szybszym tempie powodując delikatną zmianę wysokości dźwięku.
Aby otrzymać różne (parzyste) wariacje w modulacji wysokości dźwięku, linia opóźniająca jest modyfikowalna przy pomocy tablicy „Wayetable" Jest to podobny efekt do Chóralnego tylko bezpośredni sygnał nie jest miksowany z linią opóźnienia wyjścia.

Ten efekt jest często mylony z „tremolo” gdzie amplituda jest rozróżniana przez LFO. Tremolo i vibrato mogą być mieszane ze sopla z różnym w czasie LPF, aby wyprodukować efekt obracającego mówcy.




**Stereo chorus efekt.**

Efekt ten jest osiągnięty przez płynny ruch tam i z powrotem na oba kanały stereo ( powstaje w ten sposób wrażenie ruchu dźwięku w przestrzeni). Efekt ten może być tworzony też przez wysyłanie niezmienionego sygnału wejściowego na jeden wyjściowy kanał stereo i chóralny efekt w przeciwległym kanale.



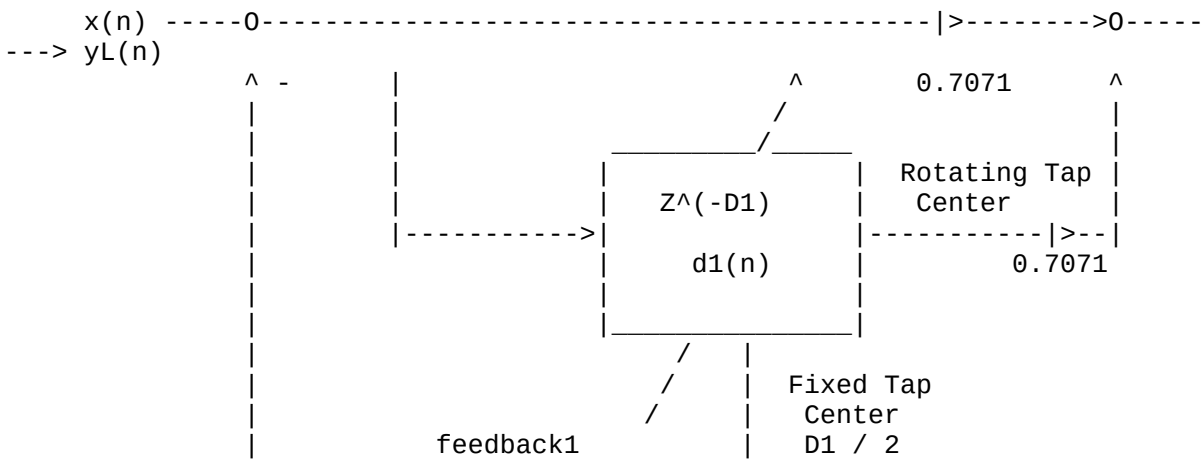

## Przykład Stereo Chorus Efekt

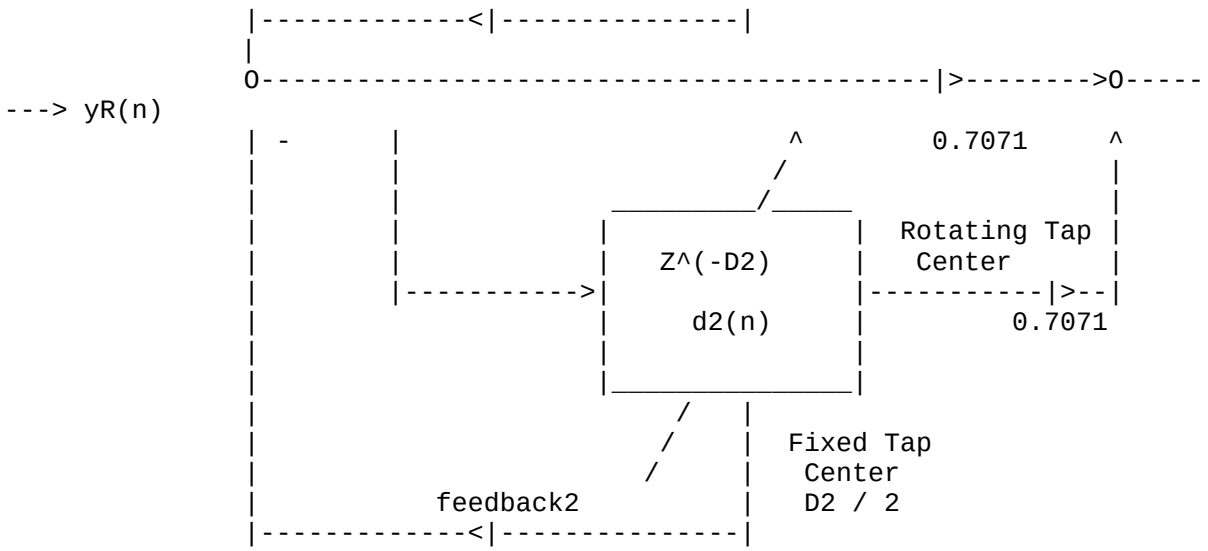
W tym ćwiczeniu użyta została Interpolacja Liniowa wraz z całkowitym (nie cząstkowym) opóźnieniem próbki.

Z próbkowaniem rzędu 48kHz Interpolacja Liniowa jest adekwatna dla większości instrumentów. Równania I/O :

$$yL(n) = 1.0 * x(n) + 0.7071*x(n - d1(n)) - 0.7071*x(n - D1/2)$$

$$yR(n) = 1.0 * x(n) + 0.7071*x(n - d2(n)) - 0.7071*x(n - D2/2)$$






Efekt symuluje wrażenie wielu instrumentów muzycznych grających na raz ten sam utwór. Efekt ten jest uzyskiwany poprzez losowe lub sinusoidalne zmiany w amplitudzie i opóźnieniu.

Dla każdej próbki algorytm:

Zachowuje próbki wejściowe s0 w dwóch liniach opóźniających
modyfikuje tablice sygnału (jeśli potrzeba)
generuje losowe opóźnienie, d = D * (0.5 + randnum(fc*t))
s1 = sample1 = tap(D, w1, p1, d)
s2 = sample2 = tap(D/2)
y = a0 * s0 + a1 * s1 - af * s2

**Zadania do wykonania**

- Podłącz mikrofon do Mic In i głośniki do Ch2 (Line Out) w 21161 Ezkit Lite
- Otwórz projekt "Stereo_Chorus_Effect.dpj" w VisualDSP++ Integrated Development Environment (IDE).
- W menu "Project" wybierz "Build Project".
- Otwórz sesję ADSP21161 EZ-KIT Lite w debuggerze VisualDSP++.
- Załaduj "Stereo_Chorus.dxe"
- Uruchom projekt i zacznij mówić do mikrofonu by usłyszeć efekt.
- Przyciskiem IRQ2 można zmieniać ustawienia:
    * IRQ2 bez FLAG1 zmienia ustawienia tempa i szerokości dźwięków
    * IRQ2 wraz z FLAG1 (jako pierwszym)zmienia ustawienia pętli zwrotnej
- FLAG3 pomija efekt.
- Zapoznaj się z kodem programu.
- Zmiana parametrów sprzężenia

feedback_settings_1:

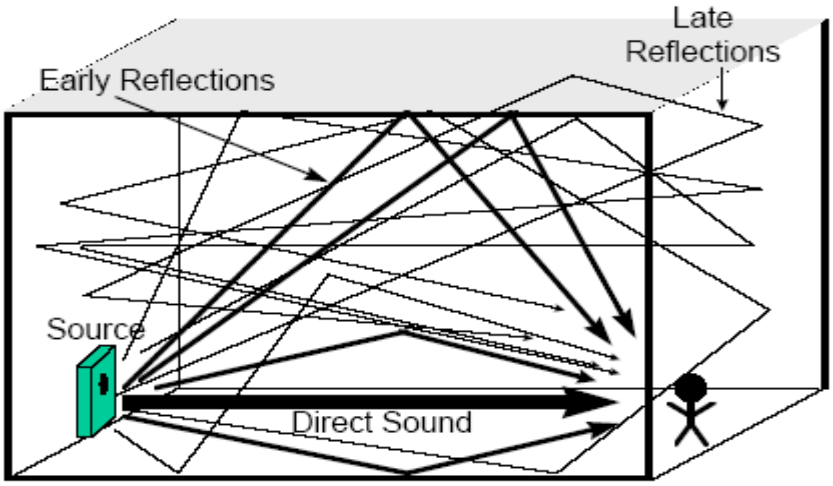
```
/* no feedback */
r14 = 0x00000000;        DM(feedback_gainL) = r14
r14 = 0x00000000;        DM(feedback_gainR) = r14;
ustat1=DM(IOFLAG);
bit set ustat1 0x3E;
bit clr ustat1 0x01;
```

```
        dm(IOFLAG)=ustat1;
   jump exit_chorus_feedback;

proponowane zmiay wartości sprzezenia
        r14 = 0x20000000;       DM(feedback_gainL) = r14;
        r14 = 0x20000000;       DM(feedback_gainR) = r14;

        r14 = 0x5A82799A;       DM(feedback_gainL) = r14;
        r14 = 0x5A82799A;       DM(feedback_gainR) = r14;
```

-Zmiana opóźnienia  efektu –modyfikacja szerokości sygnału oraz jego tempa

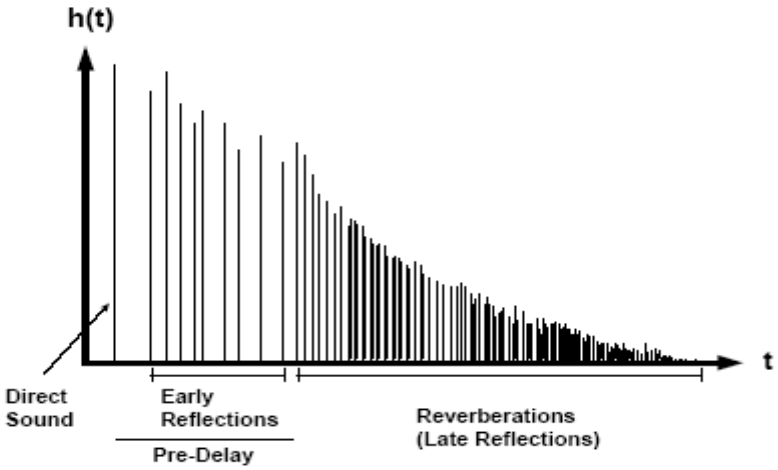
```
delay_settings_1:
        r14 = 80;           DM(sweep_rate) = r14;
        r14 = -1;           DM(sweep_widthL) = r14;
        r14 = -1;           DM(sweep_widthR) = r14;
        ustat1=DM(IOFLAG);
        bit clr ustat1 0x3E;
        bit set ustat1 0x01;
        dm(IOFLAG)=ustat1;
   jump done_depth_change;
```

## 5. Cyfrowe algorytmy odbicia dla symulacji dużych akustycznych przestrzeni



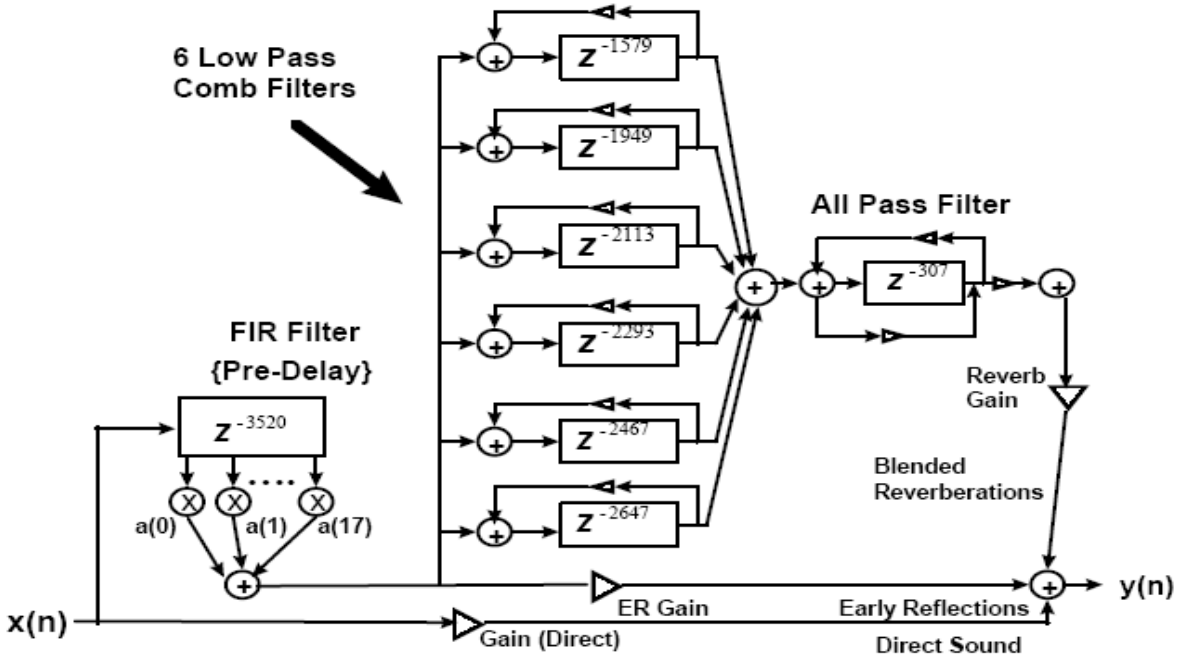

Efekt „echa” jest zasymulowany jako odbicie w dużej sali np. koncertowej. W wyniku powstania wielu opóźnień, powtórzeń ucho ludzkie nie jest w stanie odróżnić różnic miedzy tymi opóźnieniami. Powtórzenia te zostają zmieszane w sygnał ciągły. Emitowany dźwięk w wyniku wielu odbić w różnych kierunkach dociera do odbiorcy z różnych kątów z różnymi opóźnieniami. Odbicia najczęściej zachodzą w zamkniętych przestrzeniach i są mocniejsze dla twardych powierzchni. W odbiciach dźwięku możemy wyróżniamy trzy fazy : bezpośredni dźwięk , wczesne odbicie i zmieszanie sygnału echa (późne odbicie) .

**Odpowiedz impulsowa dla odbić w dużej przestrzeni**

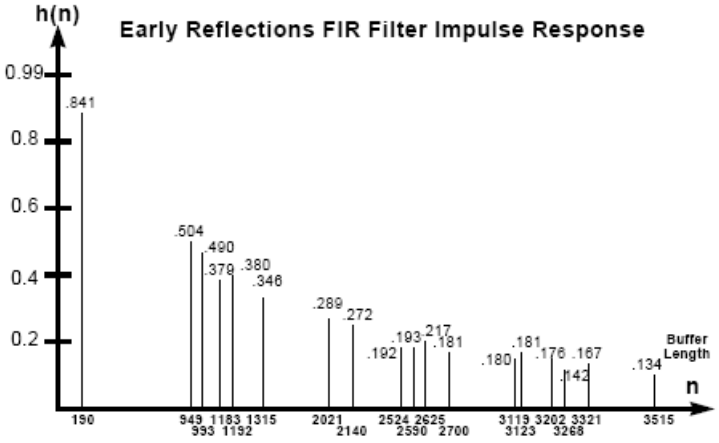



W takim otoczeniu pierwszy wyraźny impuls wyłapywany przez użytkownika możemy nazwać wczesnym odbiciem. Wczesne odbicia są skutkiem pierwszego odbicia „zwrotnego" sygnału od najbliższej powierzchni. Następne odbicia są już nie rozpoznawalne dla ucha ze względu na małe interwały czasowe. W cyfrowym odbiciu typowo sygnał przetwarzany jest przez wiele filtrów opóźniających i dodaje z wartością sygnału wczesnego odbicia. Różnymi parametrami do rozważenia są decay time, presense (sygnał wyjściowy/odbicia), kontrola tonów (basy, soprany).W celu uzyskania lepszego efektu zasugerowano 2 podejścia: zastosowanie pięciu filtrów wszechprzepustowych połączonych kaskadowo, drugie równoległe połączenie czterech filtrów grzebieniowych a ich wyjścia zostały by podane na dwa filtry wszechprzepustowe połączone kaskadowo.
Moorer w wyniku badań ustalił, że sygnały o dużej częstotliwości maja większą skłonność do odbić niż o małej. Zastosowano tu filtr dolnoprzepustowy grzebieniowy dla każdego poziomu w celu zwiększenia gęstości uzyskanej odpowiedzi, co pociągnęło za sobą potrzebę użycia sześciu filtrów dolnoprzepustowych grzebieniowych, a następnie podanie ich wyjść na filtr wszechprzepustowy przed uzyskaniem końcowego wyniku. Zarekomendowano również badanie wczesnych odbić za pomocna filtra FIR tapped-delay w celu uzyskania jak najlepszego dźwięku w sali koncertowej. Taka technika umożliwia dodanie początkowego opóźnienia rzędu 80ms. Moorer dobrał tak współczynniki filtru aby uzyskać dziewiętnaście wczesnych odbić, ale i tak nie udało mu się uzyskać idealnego dźwięku – w niektórych przypadkach był on szorstki.

**Digital Reverberation Structure**




Jest to struktura zastosowana przy próbkowaniu z częstotliwością 44.1kHz z zastosowaniem sześciu filtrów dolnoprzepustowych i wyjściem podanym na filtr wszechprzepustowy. Dla uzyskania dobrego dźwięku DSP musi zapewnić duże opóźnienie nie tylko dla filtrów grzebieniowych ale również dla buforów wczesnego odbicia ER. Każdy filtr ma różna długość linii opóźniającej. Czas opóźnienia odbicia zależy od rozmiaru bufora opóźniającego oraz częstotliwości próbkowania.




**Example Reverb Specifications for a Large Auditorium response at 44.1 kHz Sampling Rate**

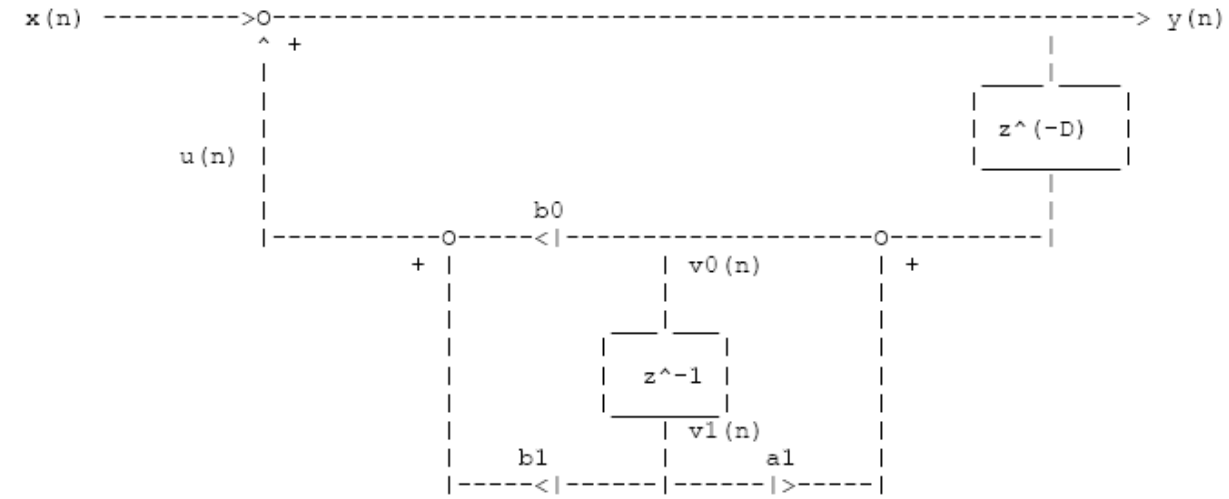
| *Delay Line Buffer Length* | | *Time Delay* |
|---|---|---|
| *Comb 1* | *1759* | *40 ms* |
| *Comb 2* | *1949* | *44 ms* |
| *Comb 3* | *2113* | *48 ms* |
| *Comb 4* | *2293* | *52 ms* |
| *Comb 5* | *2467* | *56 ms* |
| *Comb 6* | *2647* | *60 ms* |
| *Early Reflections* | *3520* | *80 ms* |
| *All-Pass Filter* | *307* | *7 ms* |

**Struktura filtrów grzebieniowych dolno i wszechprzepustowych.**

Filtry te stosuje się w celu zwiększenia gęstości echa, a co za tym idzie poprawienia jakości odbieranego przez człowieka sygnału. Dzieje się to za pomocą uzyskania odpowiednich opóźnień. Zastosowanie filtru dolnoprzepustowego zapewnia usunięcie metalicznego dźwięku i skraca czas odbicia sygnałów o dużych częstotliwościach. Filtr wszechprzepustowy zmienia fazę co pozwala uzyskanie dźwięku zbliżonego do realnego.

Struktury filtrów:

Low Pass IIR Comb Filter Structure:




Allpass Filter Structure:

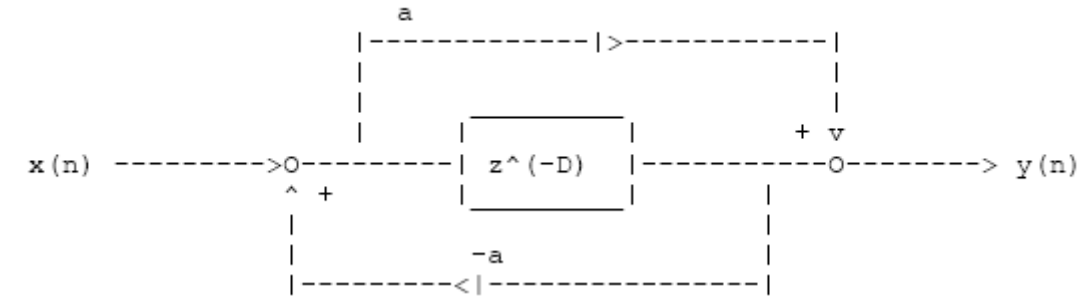
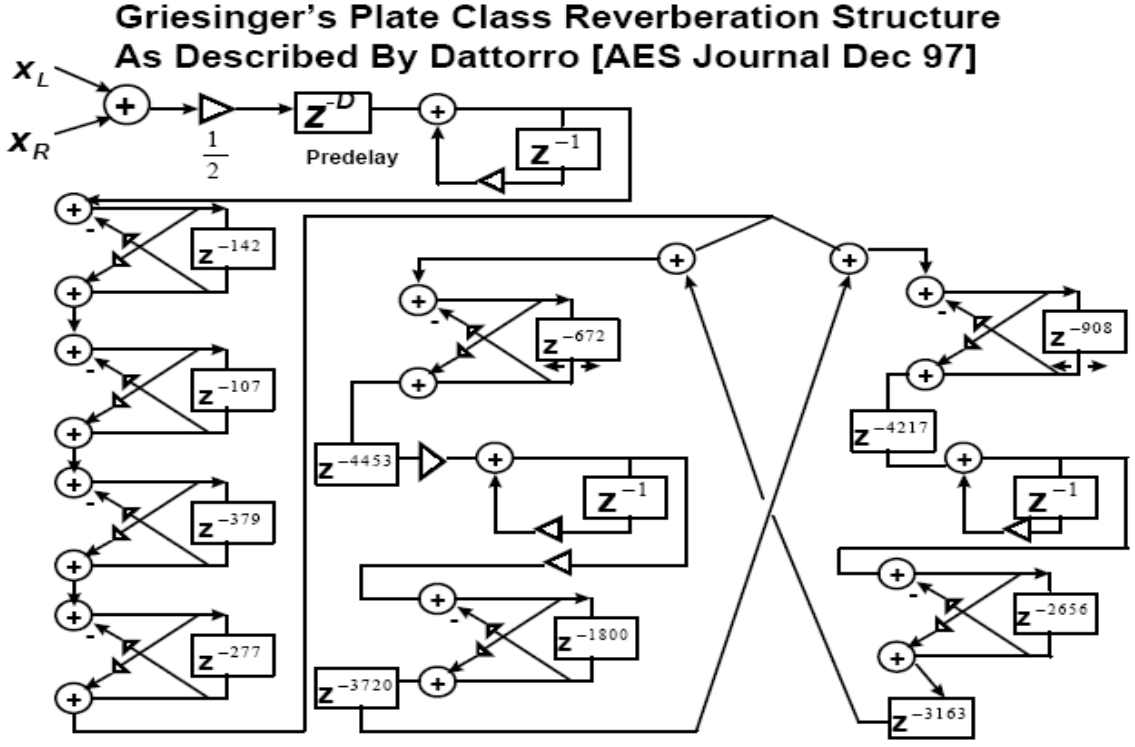



Plate Reverb To efekt opisany przez Dattorro, oparł się on na filtrach grzebieniowych oraz wykorzystał strukturę kratową.

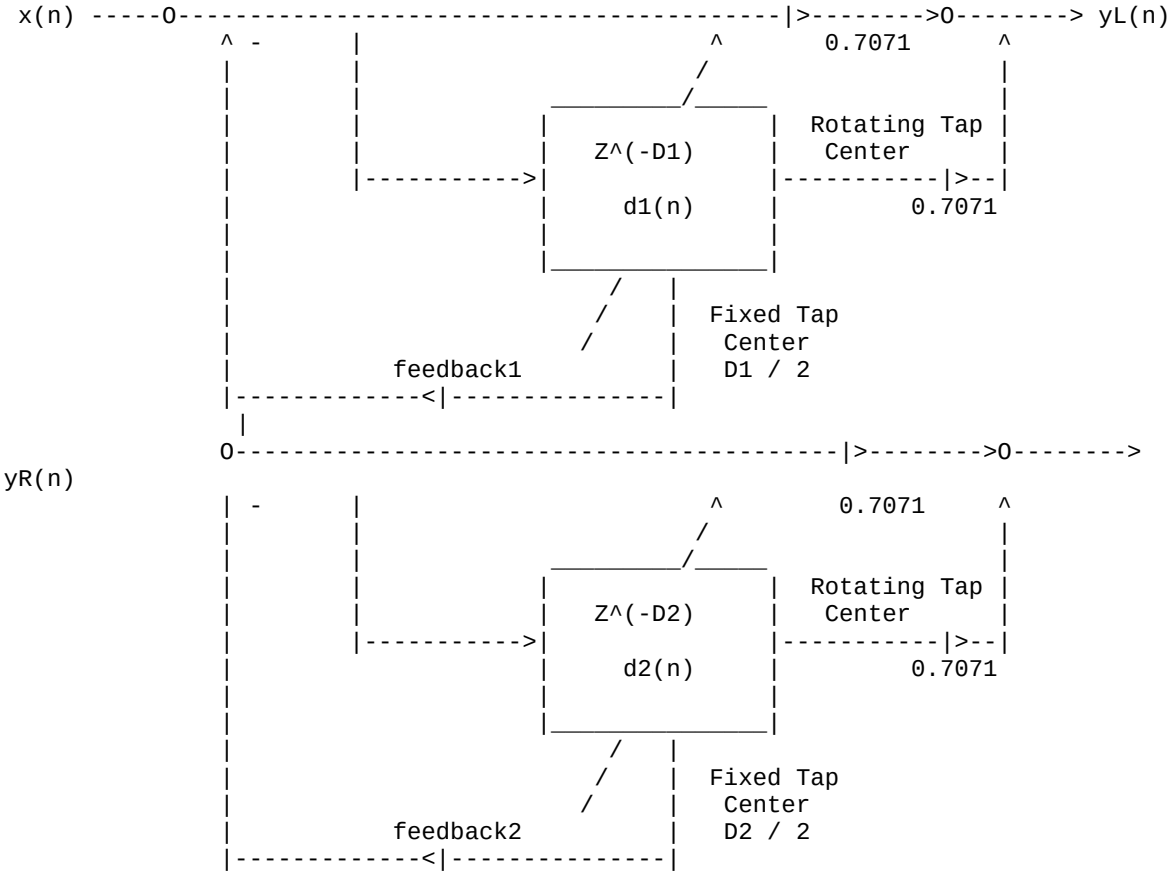


## Przykład Griesinger & Stereo Chorus

Bazą tego projektu jest przykład "talkthrough" (przeniesienie wejścia na wyjście bez modyfikacji) dla procesora 21161. Funkcja modyfikująca jest wklejona w funkcję Talkthrough.

W tym ćwiczeniu użyta została Interpolacja Liniowa wraz z całkowitym (nie cząstkowym) opóźnieniem próbki.
Z próbkowaniem rzędu 48kHz Interpolacja Liniowa jest adekwatna dla większości instrumentów.
Równania I/O:

yL(n) = 1.0 * x(n) + 0.7071*x(n - d1(n)) - 0.7071*x(n - D1/2)
yR(n) = 1.0 * x(n) + 0.7071*x(n - d2(n)) - 0.7071*x(n - D2/2)




Efekt symuluje wrażenie wielu instrumentów muzycznych grających na raz ten sam utwór podczas prawdziwego koncertu. Efekt ten jest uzyskiwany poprzez losowe lub sinusoidalne zmiany w amplitudzie i opóźnieniu.
Dla każdej próbki algorytm:

-zachowuje próbki wejściowe s0 w dwóch liniach opóźniających
-modyfikuje tablice sygnału (jeśli potrzeba)
-generuje losowe opóźnienie, d = D * (0.5 + randnum(fc*t))
s1 = sample1 = tap(D, w1, p1, d)
s2 = sample2 = tap(D/2)
y = a0 * s0 + a1 * s1 - af * s2

**Zadania laboratoryjne**

- Podłącz mikrofon do Mic In i głośniki do Ch2 (Line Out) w 21161 Ezkit Lite
- Otwórz projekt "QuadReverbStereo_Chorus.dpj" w VisualDSP++ Integrated Development Environment (IDE).
- W menu "Project" wybierz "Build Project".
- Otwórz sesję ADSP21161 EZ-KIT Lite w debuggerze VisualDSP++.
- Załaduj "Plate_Reverb&Stereo_Chorus.dxe"
- Uruchom projekt  i zacznij mówić do mikrofonu by usłyszeć efekt.
- Użyj przycisków, by zmienić efekt:
    * IRQ2 zmienia zanikanie odbić
    * IRQ1 zmienia ustawienia prawego/lewego miksera odbić
    * IRQ0 bez FLAG1 zmienia ustawienia tempa i szerokości dźwięków

```
delay_settings_1:
        r14 = 80;          DM(sweep_rate) = r14;
        r14 = -1;          DM(sweep_widthL) = r14;
        r14 = -1;          DM(sweep_widthR) = r14;
        ustat1=DM(IOFLAG);
        bit clr ustat1 0x3E;
        bit set ustat1 0x01;
        dm(IOFLAG)=ustat1;
   jump done_depth_change;
```

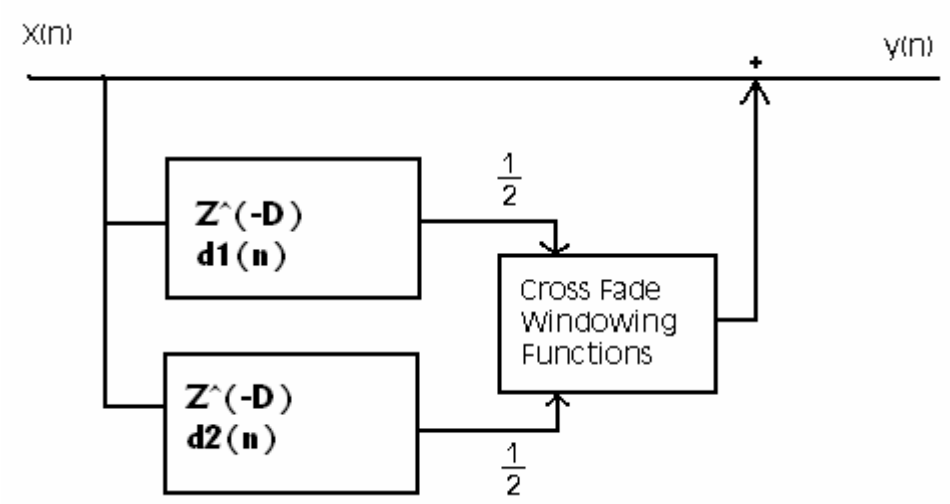
- FLAG3 pomija efekt.
- FLAG2 wyłącza efekt chóralny, przy zachowaniu efektu odbić.
- Zapoznaj się z kodem programu.
- Zmień wartości sprzężenia

```
feedback_settings_1:
        /* no feedback */
        r14 = 0x00000000;        DM(feedback_gainL) = r14;
        r14 = 0x00000000;        DM(feedback_gainR) = r14;
        ustat1=DM(IOFLAG);
        bit set ustat1 0x3E;
        bit clr ustat1 0x01;
        dm(IOFLAG)=ustat1;
   jump exit_chorus_feedback;.
```

# Jeśli pozostało ci trochę czasu przebadaj jeszcze inne efekty akustyczne (2), np.:

## 1.Pitch shifter

Interesującym i szeroko stosowanym efektem jest zmiana wysokości tonu instrumentu lub głosu. Algorytmem służącym do zaimplementowania przesuwnika tonu jest efekt chóralny lub wibrujący. Funkcja chóru używana jest wtedy, gdy użytkownik chce połączyć sygnał przesunięty z oryginalnym. Wibrujący służy do odtworzenia tylko sygnału zmodyfikowanego, co jest często wykorzystywane w stacjach TV by zniekształcić głos osoby, która życzy sobie pozostać anonimową.



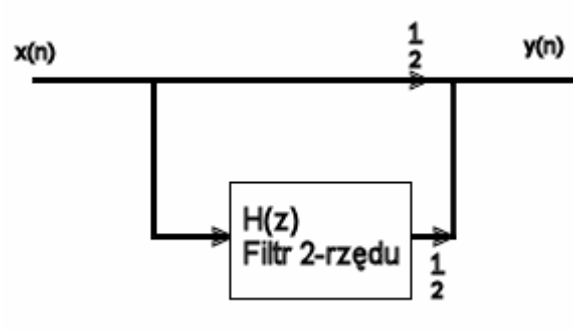

Pitch shifter korzysta z przebiegu piłokształtnego by uzyskać efekt liniowego opadania i dodawania próbek do przetwarzanego fragmentu z bufora wejściowego. Nachybenie zbocza piły wraz z opóźnieniem steruje uzyskanym efektem.

Słyszalnym skutkiem ubocznym stosowania chóru dwuinstrumentowego z jednąlinią opóźniającą są trzaski powstające przy każdym przejściu wskaźnika opóźnienia przez wskaźnik sygnału, kiedy próbki się dodają lub znoszą. Dzieje się tak z powodu, że wskaźnik wyjściowy porusza się po buforze szybciej/wolniej niż wskaźnik wejściowy, a więc zajdzie taka sytuacja, gdy jeden wyprzedzi drugi. By usunąć lub przynajmniej zredukować ten nieporządany efekt, ta wersja programu używa techniki cross-fading pomiędzy dwoma zmiennymi buforami opóźnienia z funkcją okna, więc kiedy dwa wskaźniki są blisko siebie, następuje przejście przez zero, przez co unika się niechcianego "szczeknięcia". Dla wyższych wartości przesunięcia można zauważyć "harczenie" będące rezultatem rozsunięcia się w fazie linii opóźniających, które powoduje okresowy zanik pewnych częstotliwości. Istnieją metody zapobiegania takiemu zjawisku, jednakże nie zostały one zaimplementowane w tym ćwiczeniu.

Dla każdej próbki wejściowej, algorytm:
zachowuje próbkę s0 w dwóch liniach opóźniających
modyfikuje tablicę (jeśli trzeba) i uaktualnia wartości okna triangulacyjnego
generuje losowe opóźnienie, dx = Dx * (0.5 + sawtooth(fc*t))
s1 = sample1 = tap(D, w1, p1, d1)
s2 = sample2 = tap(D, w2, p2, d2)
y = a0 * s0 + a1 * s1 + a2 * s2

# 2 Phaser Effect

Ten popularny wśród muzyków efekt jest uzyskiwany poprzez przepuszczenie sygnału przez wąski filtr pasmowy i połączenie tak przetworzonego sygnału z oryginałem. Częstotliwość pasma jest następnie zmieniana za pomocą oscylatora niskiej częstotliwości. Duże przesunięcia fazowe występujące w okolicach częstotliwości pasma łączą się z sygnałem oryginalnym i powodują wygaszenia lub wzmocnienia, co zmienia spektrum częstotliwości. W tym przypadku do wejścia filtra dodawana jest także część sygnału wyjściowego y(n) dla wzmocnienia efektu.




General Design Equation for 2nd-Order Single Notch Filter:

$$H(z) = \frac{b*(1-2\cos(w0)z^{-1}+z^{-1})}{1-2\cos(w0)z^{-1}+(2b-1)z^{-2}}$$

Notch Filter Specifications:

Design width Df = 300

$$D_w = \frac{2\pi * Df}{f_s} = 2\pi\frac{300Hz}{48kHz} = 0{,}0125\pi = 0{,}039269908$$

$$b_0 = \frac{1}{\left(1+\tan\left(\frac{D_w}{2}\right)\right)} = 0{,}980747025$$

Sweep frequency fsweep = 1 Hz.

w1 (f1) = center notch frequency when sin(angle) = 0
w2 (f2) = max swing notch frequency when sin(angle) = +1 or -1

Notch frequency # 1 varies sinusoidally from 200 Hz to 800 Hz:

$f_0 = 500 + 300\sin(2\pi f_{sweep} t)$ (in Hz)

$w_0 = 0{,}0208\pi + 0{,}0125\pi\sin(2\pi f_{sweep} t)$ (in rads/sample)

Notch frequency # 2 varies sinusoidally from 200 Hz to 1800 Hz:

$f_0 = 1000 + 800\sin(2\pi f_{sweep} t)$ (in Hz)

$w_0 = 0{,}01467\pi + 0{,}0333\pi\sin(2\pi f_{sweep} t)$ (in rads/sample)

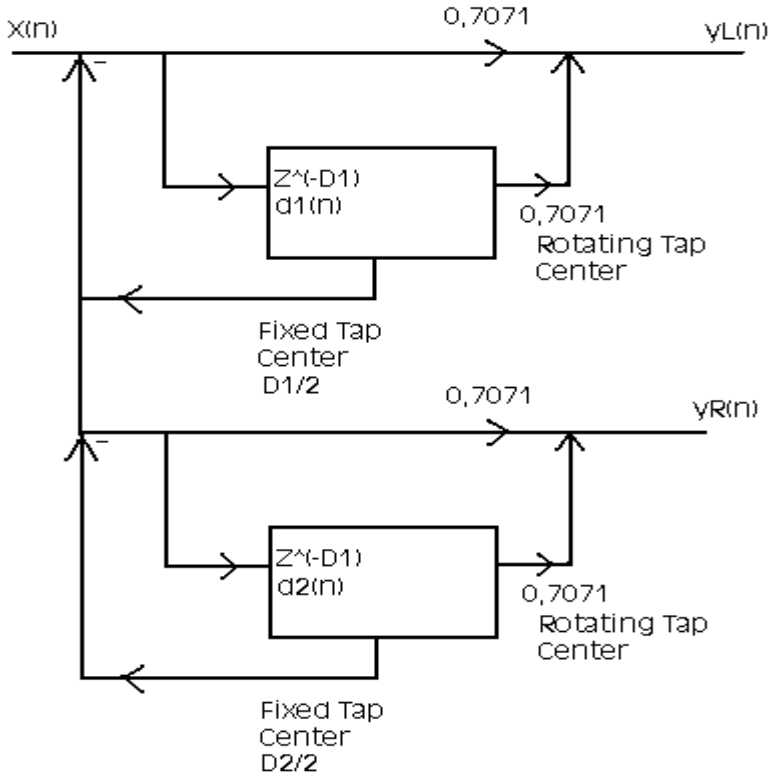
Notch frequency # 3 varies sinusoidally from 400 Hz to 3600 Hz:

$f_0 = 2000 + 1600\sin(2\pi f_{sweep} t)$ (in Hz)

$w_0 = 0{,}08333\pi + 0{,}06667\pi\sin(2\pi f_{sweep} t)$ (in rads/sample)

W tym ćwiczeniu do obliczania funkcji cos został użyty szereg Taylora.

Dla małych używa się przybliżenia wynikającego z rozkładu przybliżenia.




Flanging to modulowanie opóźnionej o kilka milisekund kopii sygnału wejściowego i dodanie jej do sygnału oryginalnego. Spowoduje to przesunięcia fazowe i "rozjeżdżanie się" sygnału. Opóźnienie jest modulowane za pomocą wolnozminennej sinusoidy. Efekt ten najlepiej stosować w gitarach, bębnach i u niektórych solistów.

Dla każdej próbji algorytm:
zachowuje próbkę wejściową s0 w buforze linii opóźniającej flangera - *p = s0 = xinput

generuje losowe opóźnienie,

s1 = delayed sample = tap(D, w, p, d)
y = a0 * s0 + a1 * s1

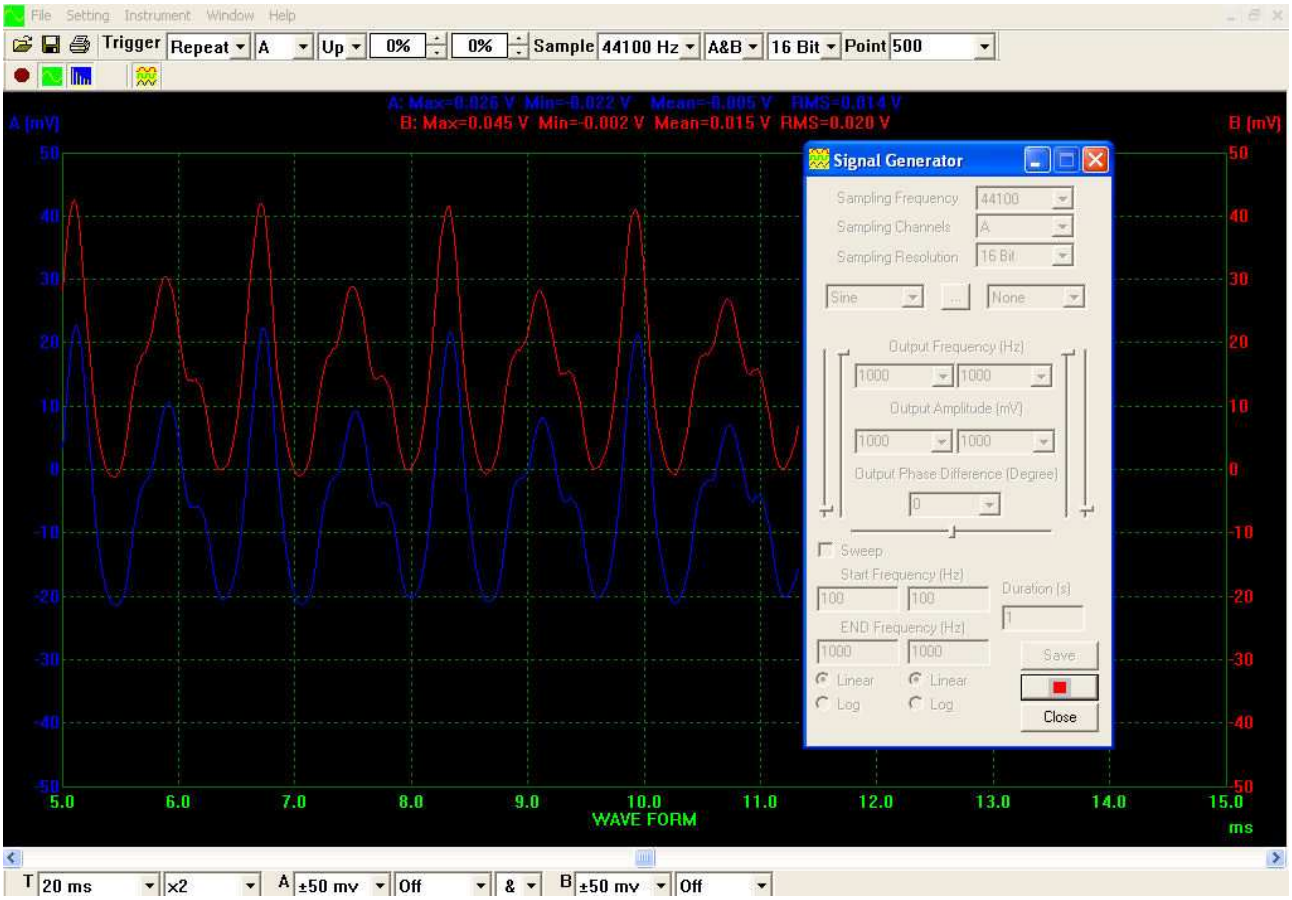
# Zadania:

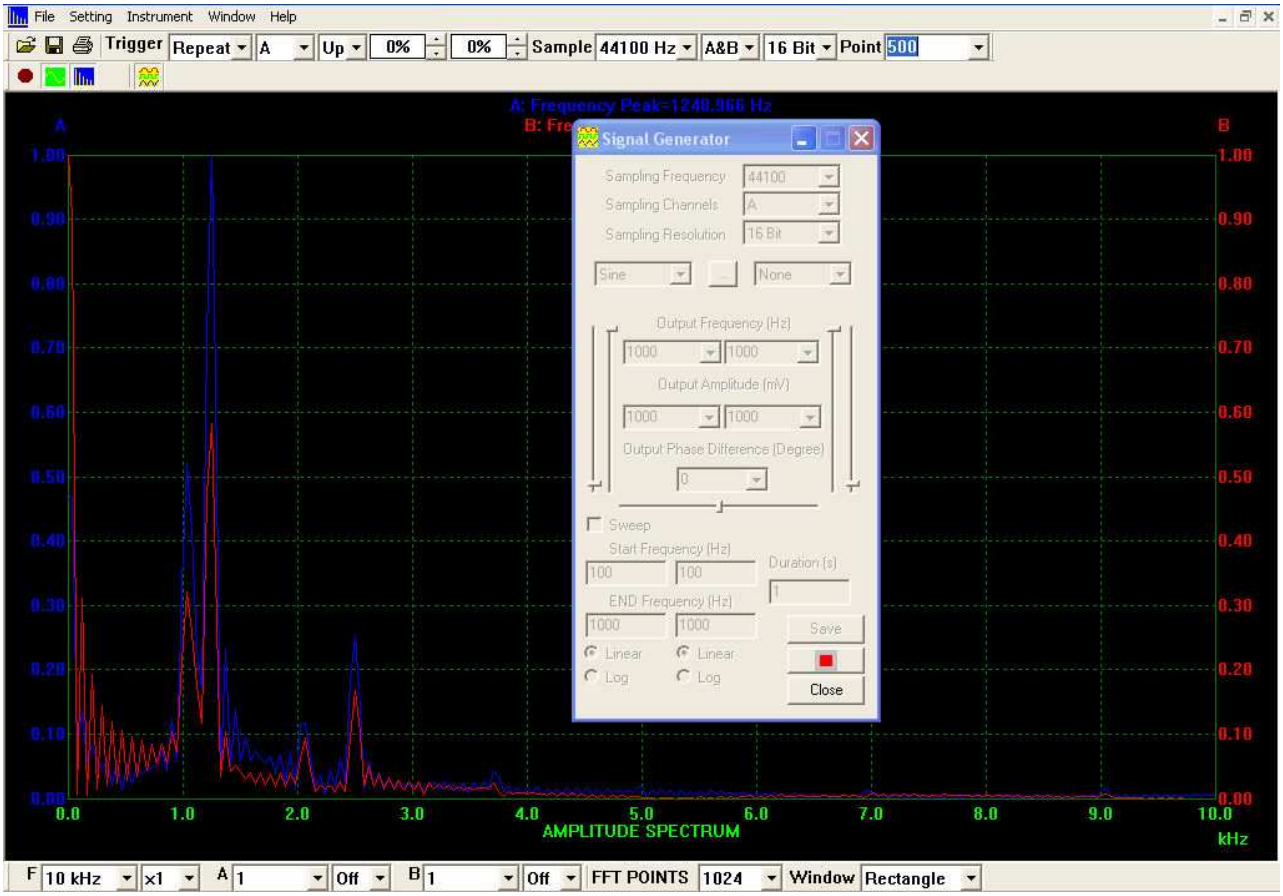
## 1) Pitch shifter

- Podłącz mikrofon do gniazda Mic In i głośniki do Ch2 (Line Out) w 21161 Ezkit Lite
- Otwórz projekt "Pitch_Shifter.dpj" w VisualDSP++ Integrated Development Environment(IDE).
(C:\Program Files\Analog Devices\VisualDSP 4.0\211xx\EZ-KITs\ADSP-21161N\Examples\ASM\Digital_Audio_Effects\Pitch_Shifter)

- W menu "Project" wybierz "Rebuild Project".
- Uruchom projekt  (Run F5) i zacznij mówić do mikrofonu by usłyszeć efekt.
- Przyciskiem IRQ2 można zmieniać ustawienia efektu. (6 wariantów widocznych na diodach)
- Przyciśnięcie FLAG3 powoduje pominięcie efektu.

- Podłącz zielone gniazdo karty muzycznej z Mic In, niebieskie gniazdo z Ch2
- Uruchom program „Scins” z pulpitu
- Włączyć „Signal Generator” z menu Instrumnet
- Wybrać przebieg sine np. z częstotliwością 1 kHz i amplitudą 1V
- Włączyć generator i analizator sygnału.





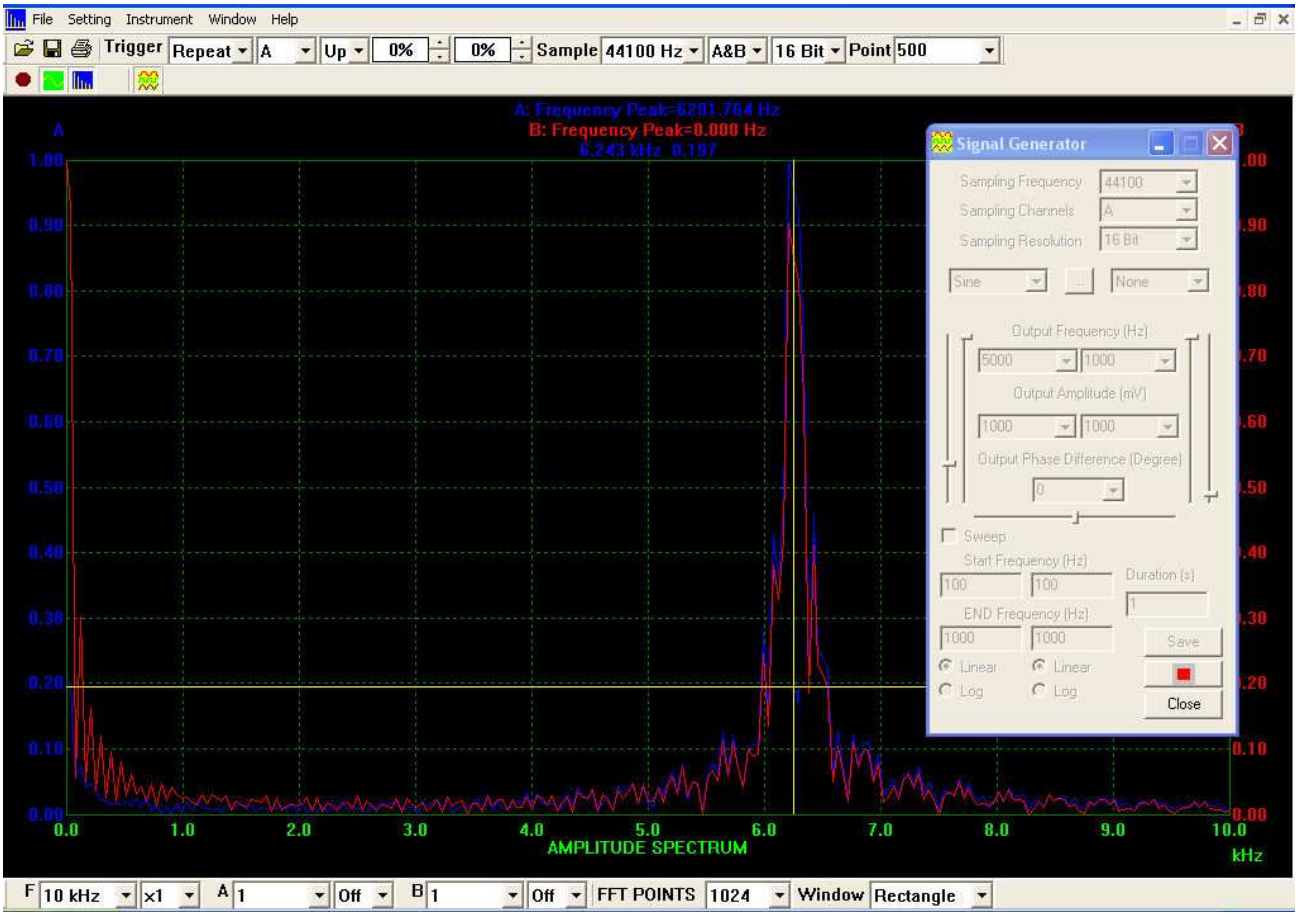

- Przeanalizować przebiegi wyjściowe na oscyloskopie i analizatorze widma

## 2) Phaser Effect

- Podłącz mikrofon do gniazda Mic In i głośniki do Ch2 (Line Out) w 21161 Ezkit Lite
- Otwórz projekt "Phaser_Effect.dpj" w VisualDSP++ Integrated Development Environment(IDE).
  (C:\Program Files\Analog Devices\VisualDSP 4.0\211xx\EZ-KITs\ADSP-21161N\Examples\ASM\Digital_Audio_Effects\Phaser Effect)

- W menu "Project" wybierz "Rebuild Project".
- Uruchom projekt (Run F5) i zacznij mówić do mikrofonu by usłyszeć efekt.
- Przyciskiem IRQ2 można zmieniać ustawienia efektu. (6 wariantów widocznych na diodach)
- Przyciśnięcie FLAG3 powoduje pominięcie efektu.

- Podłącz zielone gniazdo karty muzycznej z Mic In, niebieskie gniazdo z Ch2.
- Uruchom program „Scins” z pulpitu.
- Włączyć „Signal Generator” z menu Instrumnet.
- Wybrać przebieg sine np. z częstotliwością 5 kHz i amplitudą 1V.
- Włączyć generator i analizator sygnału.




- Przeanalizować przebiegi wyjściowe na analizatorze widma.

# Literatura:


1) 7056820721065L_Audio_Tutorial.pdf z tutorial.
2) Pliki tekstowe : Stereo Flanger Effect.txt, Pitch Shifter 2 Part Harmony with Crossfade.txt, Stereo Flanger Effect.txt
3) Internet : http://infinityanalog.com/analog/tone/?sub=Phaser%20Effect
   http://www.harmony-central.com/Effects/Articles/Flanging/