

Układy Cyfrowe – laboratorium

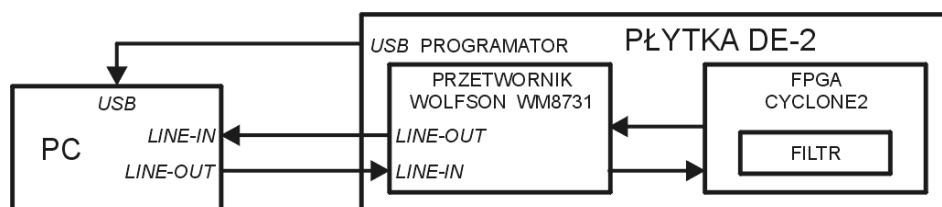
Przykład realizacji ćwiczenia nr 8 (wersja 2015)

1. Wstęp

Komputer PC jest użyty do syntezy struktury i konfiguracji układu FPGA (Quartus2), dodatkowo służy jako generator i analizator sygnału audio (Spectrum Lab).

Analogowy sygnał audio jest wysyłany z wyjścia liniowego karty dźwiękowej komputera PC do płyty Altera DE2-115 za pomocą kabla TRS (typu cinch). Sygnał analogowy jest przetwarzany na cyfrowy w zamontowanym na płycie przetworniku analogowo-cyfrowym. Dane w postaci cyfrowej przechodzą przez projektowany przez użytkownika filtr cyfrowy w układzie FPGA i wychodzą na wejście przetwornika cyfrowo analogowego. Następnie w postaci analogowej są przesyłane do komputera PC drugim kablem TRS na wejście liniowe karty dźwiękowej komputera PC.

Kabel USB służy do wysyłania z komputera PC danych konfiguracyjnych układ FPGA przez urządzenie USB-Blaster za pomocą oprogramowania Altera QuartusII.



1.1. Konfiguracja przetwornika analogowo-cyfrowego

Celem tego zadania jest poprawne skonfigurowanie przetwornika analogowo-cyfrowego Wolfson WM8731 zamontowanego na płycie DE2-115.

UWAGA! Przed przystąpieniem do wykonania kolejnych zadań **należy wgrać** do układu FPGA konfigurację z zaimplementowanym interfejsem I2C, przez który zostaną wysłane dane ustawiające przetwornik w pożądany tryb pracy. **Wyłączenie zasilania powoduje ustawienie domyślnych parametrów przetwornika!** Przekonfigurowanie układu FPGA do realizacji filtra cyfrowego, nie zmienia wcześniej ustawionych parametrów przetwornika.

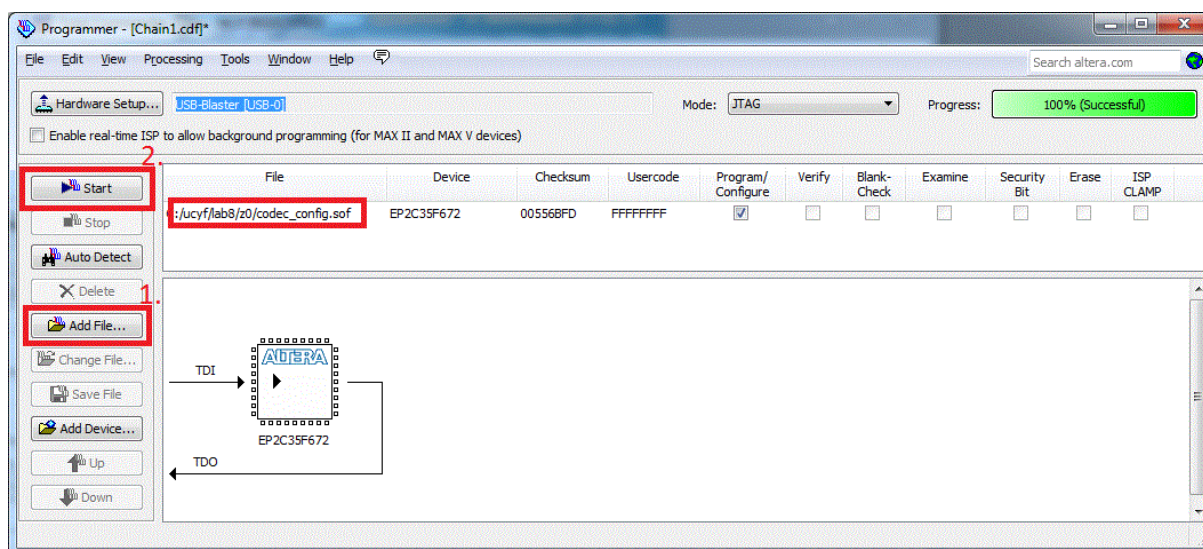
W celu przesłania danych konfiguracyjnych przetwornik AD/DA należy:
1. Upewnić się czy poprawnie podłączony jest kabel USB do złącza USB-Blaster i włączyć zasilanie płyty DE2-115.

2a. Uruchomić skrypt *prog.bat* w katalogu *./lab8/z0/*

Zawartość pliku *prog.bat*:

```
>c:\altera\13.0sp1\quartus\bin\quartus_pgm -c usb-blaster -m jtag
-o p;codec_config.sof
```

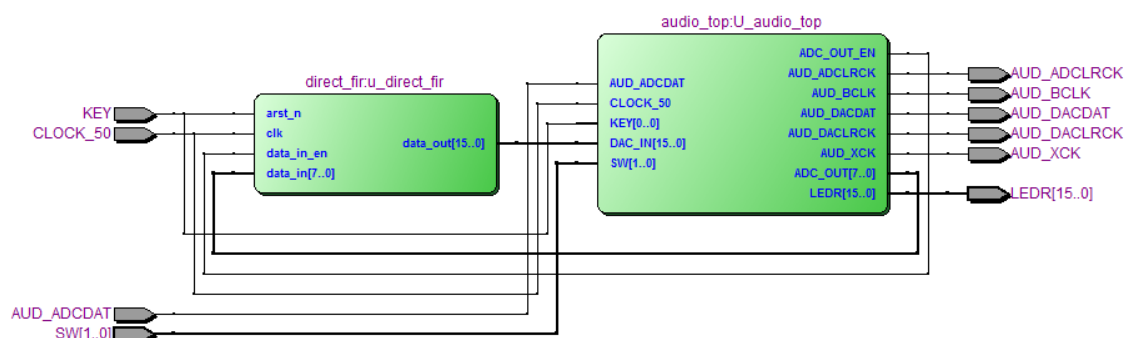
2b. Lub za pomocą modułu programatora w Quartus2 otworzyć plik *codec_config.sof* i zaprogramować układ



2. Opis struktury filtru FIR w języku VHDL

2.1. Założenia wstępne

Projekt składa się z dwóch bloków: bloku filtru cyfrowego i bloku sterującego.



Sygnal *KEY* jest sygnałem resetu asynchronicznego i przypisany jest do przycisku *KEY0* na płycie DE2-115. Przełącznik *SW[0]=0* służy do ominięcia filtru na drodze przetwarzanego sygnału, dla pozycji *SW[0]=1* filtr jest włączony. Na diodach *LEDR* można obserwować poziom sygnału.

Protokół danych wychodzących i wchodzących do bloku *audio_top*:

- próbki dźwięku wychodzące z modułu *audio_top* są w formacie U2 i mają zakres wartości $(-1,1)$ zapisany na 8 bitach,

- próbki dźwięku wychodzące z modułu *audio_top* pojawiają się z częstotliwością 48 kHz, obecność nowej wartości na linii danych *ADC_OUT* jest sygnalizowana wysokim stanem sygnałem *ADC_OUT_EN*,
- próbki dźwięku wchodzące do modułu *audio_top* są pobierane z częstotliwością 48 kHz, próbki są zatrzaskiwane w rejestrach modułu gdy sygnał *ADC_OUT_EN* jest w stanie wysokim,
- próbki dźwięku wchodzące do modułu *audio_top* są w formacie U2 i mają zakres wartości $(-1,1)$ zapisany na 16 bitach.

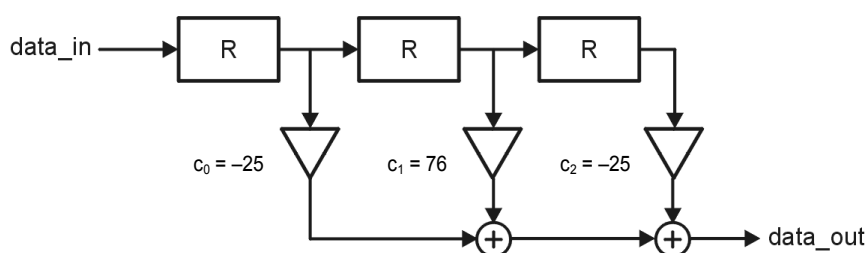
2.2.

Zadanie 1: zrealizować filtr o zadanych współczynnikach

Niech macierz filtru ma postać: $c = [-0,2; 0,6; -0,2]$.

Zadanie 1a. Narysować schemat filtru oraz zakodować współczynniki w formacie *fix1.7* i całkowitoliczbowym.

Dla podanych współczynników filtr w postaci równoległej ma postać:



Współczynniki filtru mają postać:

$$c_0 = c_2 = -0,2 = 1.1100111_{fix1.7} = -25,$$

$$c_1 = 0,6 = 0.1001100_{fix1.7} = 76.$$

Zadanie 1b. Zapisać filtr o zadanych współczynnikach w pliku *direct_fir.vhd*. Ustawić główny moduł projektu na *direct_fir*. Następnie wstawić zakodowane współczynniki w równanie opisujące filtr. Należy zwrócić uwagę na długość tablicy rejestru przesuwającego *tap* oraz zakres pętli *for*. Przeprowadzić symulację dla modułu *direct_fir.vhd* podając na wejście jednostkowy sygnał +10, a następnie podać jednostkowy sygnał -10. Zaobserwować odpowiedzi na wyjściu filtru, sprawdzając poprawność realizacji współczynników filtru.

Wydruk *direct_fir.vhd*:

```
library IEEE;
use IEEE.std_logic_1164.all;
use ieee.numeric_std.all;
```

```

entity direct_fir is
generic(
    L      : integer := 3 -- dlugosc fitru
    );
port(
    clk          : in  STD_LOGIC; -- sygnal zegara
    arst_n       : in  STD_LOGIC; -- asynchroniczny sygnal zerowania niskim
poziomem
    data_in      : in  STD_LOGIC_VECTOR(7 downto 0); -- dane wejsciowe
    data_out     : out STD_LOGIC_VECTOR(15 downto 0); -- dane wyjsciowe
    data_in_en   : in  STD_LOGIC -- strob danych wejsciowych
    );
end entity;

architecture RTL of direct_fir is
    -- definicja typu rejestrów przechowujących wartości poprzednich próbek
    type DLY_TYPE is array(2 downto 0) of signed(7 downto 0);
    -- deklaracja sygnałów przechowujących wartości poprzednich próbek
    signal tap      : DLY_TYPE;
    signal data_sum : signed(15 downto 0);
begin
    dly_synch_proc:
    process(clk, arst_n)
    begin
        if arst_n = '0' then
            for i in 0 to 2 loop
                tap(i) <= (others => '0');
            end loop;
        elsif clk = '1' and clk'event then
            if data_in_en = '1' then
                for i in 1 to 2 loop
                    tap(i) <= tap(i-1);
                end loop;
                tap(0) <= signed(data_in);
            end if;
        end if;
    end process;

    fir_sum:    data_sum <= tap(0) * to_signed( -25,8)
                + tap(1) * to_signed( 76,8)
                + tap(2) * to_signed( -25,8);
    fir_output: data_out <= std_logic_vector(data_sum);
end architecture;

```

Wydruk *test10.do*:

```

restart -nowave -force
add wave -radix signed *
run
force clk 0 1, 1 {50ps} -r 100
force data_in_en 1 0
force arst_n 0 0
force data_in 10#0 0
run
force arst_n 1 0
force data_in 10#10 0

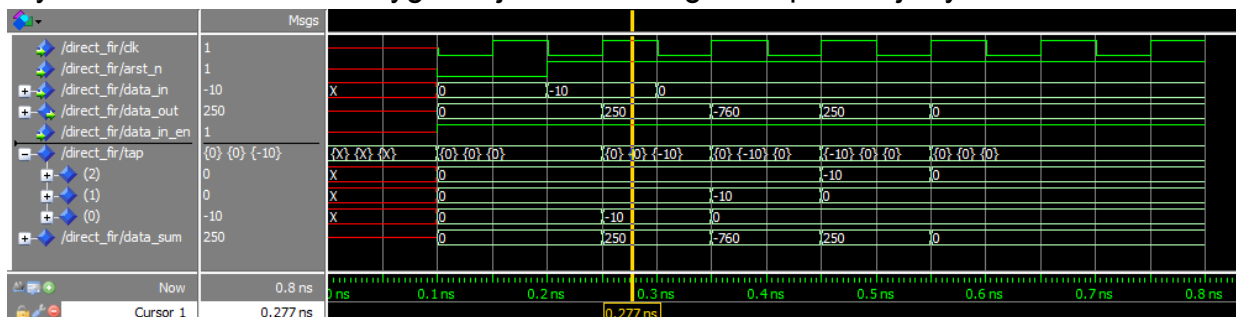
```

```

run
force data_in 10#0 0
run
run 1000

```

Wynik działania filtra dla sygnału jednostkowego -10 pokazuje rysunek:



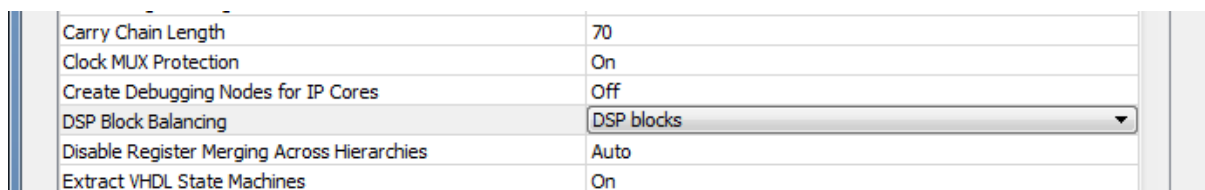
Zadanie 1c. Ustawić główny moduł projektu na *filtr_z1*.

Zmienić ustawienia kompilatora na realizację operacji mnożenia w blokach DSP.

Aby zmienić ustawienia: *Settings > Analysis & Synthesis Settings*

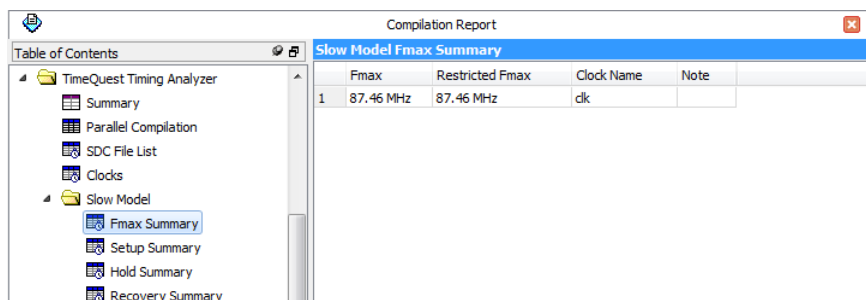
W prawym oknie wybrać opcję *More Settings...*

Odnaleźć opcję *DSP Block Balancing* i zmienić z *Auto* na *DSP blocks*.



Skompilować projekt i zanotować zużycie zasobów logicznych układu fpga (komórek *Logic Elements* = 134, pamięci *Memory bits* = 0, mnożarek *Embedded Multiplier* = 3) oraz maksymalną częstotliwość pracy f_{max} ($f = 87,46$ MHz).

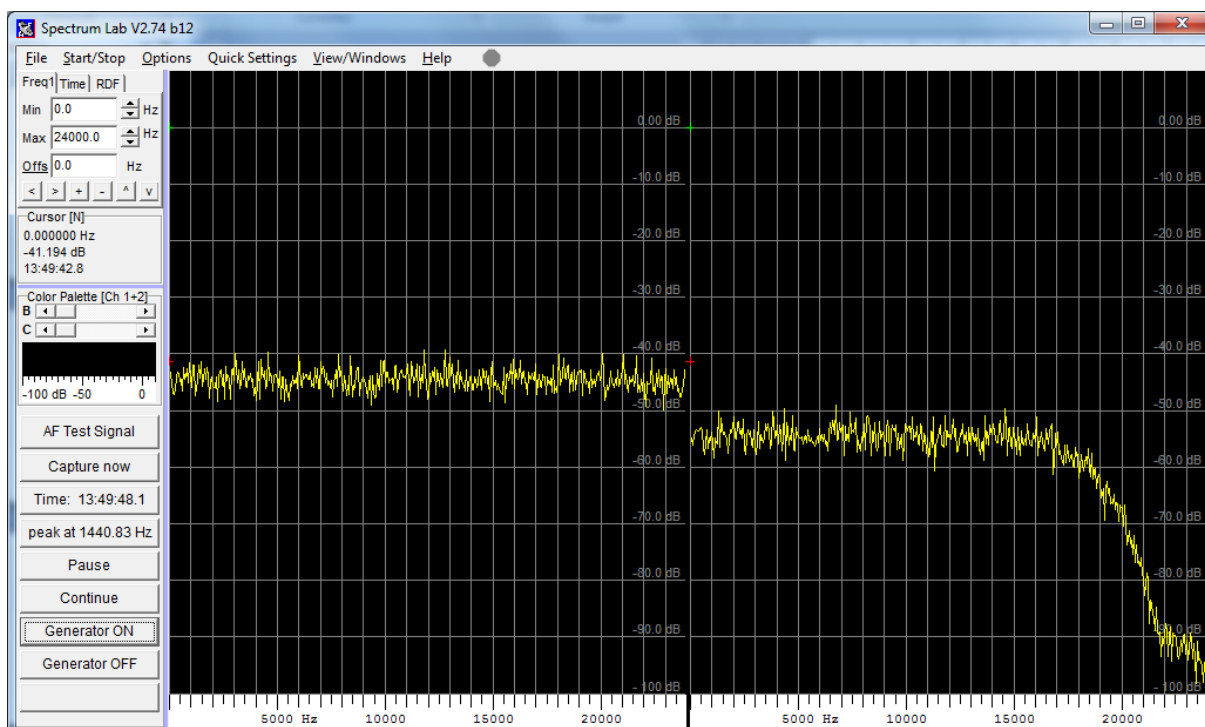
Flow Summary	
Flow Status	Successful - Sat Jan 12 14:47:08 2013
Quartus II 32-bit Version	12.0 Build 263 08/02/2012 SP 2.SJ Web Edition
Revision Name	filtr_z1
Top-level Entity Name	filtr_z1
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Total logic elements	134 / 33,216 (< 1 %)
Total combinational functions	95 / 33,216 (< 1 %)
Dedicated logic registers	100 / 33,216 (< 1 %)
Total registers	100
Total pins	26 / 475 (5 %)
Total virtual pins	0
Total memory bits	0 / 483,840 (0 %)
Embedded Multiplier 9-bit elements	3 / 70 (4 %)
Total PLLs	0 / 4 (0 %)

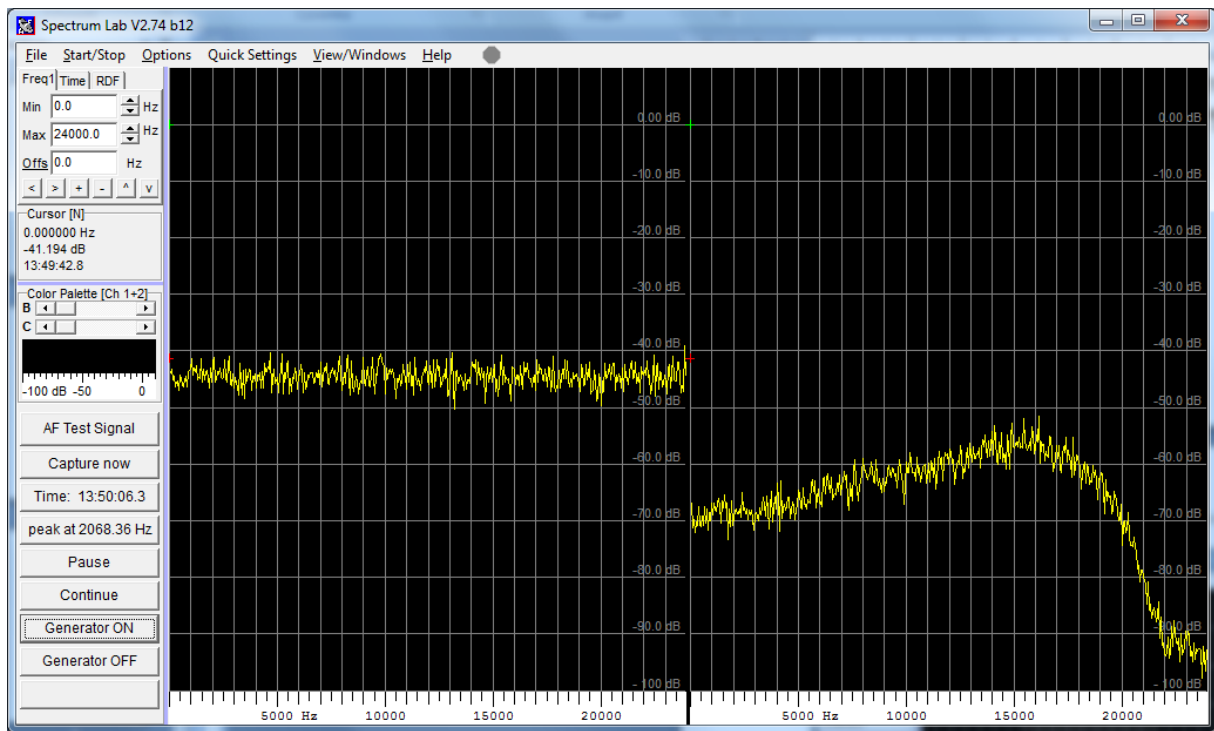


Compilation Report

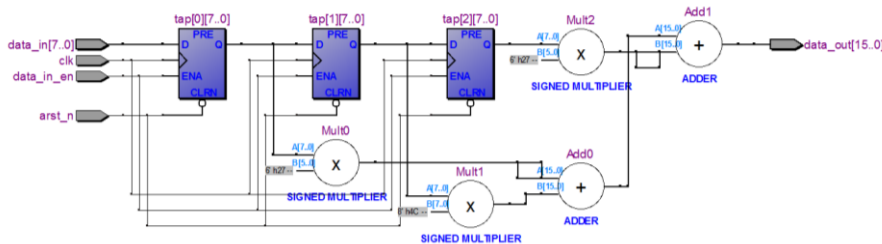
	Fmax	Restricted Fmax	Clock Name	Note
1	87.46 MHz	87.46 MHz	clk	

Zaprogramować układ fpga. Włączyć w programie Specrum Lab generator szumu i zaobserwować na wykresie działanie filtru, odpowiednio zmieniając ustawienie przełącznika SW0. Wynik działania filtru pokazuje rysunek (bez filtracji i po filtracji):





Wynik realizacji modułu filtra pokazuje rysunek:



Zadanie 1d. Zmienić ustawienia kompilatora na realizację operacji mnożenia w komórkach logicznych.

Aby zmienić ustawienia: *Settings > Analysis & Synthesis Settings*

W prawym oknie wybrać opcję *More Settings...*

Odnaleźć opcję *DSP Block Balancing* i zmienić na *Logic elements*:

Carry Chain Length	70
Clock MUX Protection	On
Create Debugging Nodes for IP Cores	Off
DSP Block Balancing	Logic Elements
Disable Register Merging Across Hierarchies	Auto
Extract VHDL State Machines	On
Extract Verilog State Machines	On
Force Use of Synchronous Clear Signals	Off

Ponownie skompilować projekt i zanotować zużycie zasobów logicznych układu fpga oraz maksymalną częstotliwość pracy f_{max} .

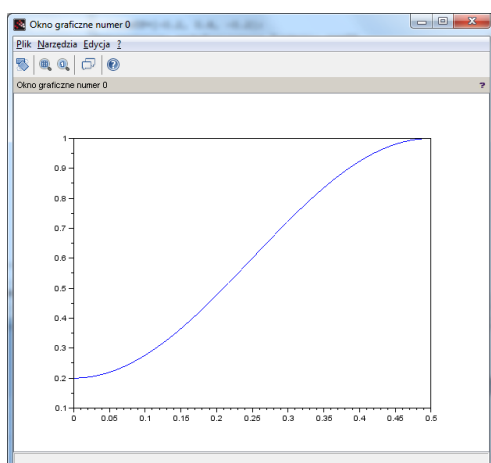
Zadanie dla zainteresowanych do domu: W programie SciLab uruchomić skrypt *z1.sci*.

Wydruk pliku *z1.sci*:

```
coeff=[-0.2, 0.6, -0.2];
[hzm,fr]=frmag(coeff,256);
plot(fr,hzm)
```

Wynik działania skryptu pokazuje wykres transmitancji filtru:

>exec *z1.sci*



Porównać wykres *plot* z oknem programu *SpectrumLab*.

2.3.

Zadanie 2: zrealizować filtr o zadanych współczynnikach w arytmetyce rozproszone (*Distributed Arithmetic*)

Niech macierz filtru ma postać: $c = [-0,2; 0,6; -0,2]$.

Zadanie 2a. ma na celu zastąpienie układów mnożących tablicą funkcji DA.

Należy wyznaczyć wiersze tablicy DA dla zadanych współczynników dla wejść 1) 0..0, 2) 1..1. (resztę tablicy wyznaczyć za pomocą programu opisanego poniżej).

Zapisując współczynniki na 8 bitach otrzymamy następującą dziesiętną reprezentację [-25, 76 , -25] liczb binarnych:

xb[2]	xb[1]	xb[0]	f(c[n],x[n])
0	0	0	$-25 \cdot 0 + 76 \cdot 0 - 25 \cdot 0 = 0$
0	0	1	$-25 \cdot 0 + 76 \cdot 0 - 25 \cdot 1 = -25$
0	1	0	$-25 \cdot 0 + 76 \cdot 1 - 25 \cdot 0 = 76$
0	1	1	$-25 \cdot 0 + 76 \cdot 1 - 25 \cdot 1 = 51$
1	0	0	$-25 \cdot 1 + 76 \cdot 0 - 25 \cdot 0 = -25$

1	0	1	$-25 \cdot 1 + 76 \cdot 0 - 25 \cdot 1 = -50$
1	1	0	$-25 \cdot 1 + 76 \cdot 1 - 25 \cdot 0 = 51$
1	1	1	$-25 \cdot 1 + 76 \cdot 1 - 25 \cdot 1 = 26$

Znamy już jeden z rozmiarów tablicy, tablica ma 3 wejścia ponieważ filtr ma 3 współczynniki. Następnie należy wyznaczyć największą liczbę na wyjściu i podać na ilu bitach można ją zakodować. W praktyce, sumujemy wszystkie współczynniki ujemne oraz sumujemy wszystkie współczynniki dodatnie. Dłuższa reprezentacja bitowa określa wielkość tablicy wyjść. W przykładzie, weźmy pod uwagę dwie liczby: najmniejszą = -50 oraz największą = 76. Liczbę -50 można zakodować na 7 bitach $100\ 1110_{U_2}$, a liczbę 76 można zakodować na 8 bitach $0100\ 1100_{U_2}$. Biorąc dłuższą liczbę binarną wyznaczyliśmy rozmiar tablicy wyjść który jest równy 8 bitów.

Następnie dla współczynników całkowitoliczbowych filtru wygenerować tablicę prawdy korzystając z oprogramowania *filter.exe* znajdującego się w katalogu *./lut_DA* i zapisać w pliku *da_table.vhd*.

Zapisać plik ze współczynnikami filtru – zawartość pliku *z2.txt*:

```
[-25, 76, -25]
```

Uruchomić program *filter.exe*, podać nazwę pliku z zapisanymi współczynnikami *z2.txt* oraz podać nazwę pliku do którego będzie wygenerowana tablica DA w formacie Berkeley z rozszerzeniem *.pla*. **Ważne! Nie podawać nazwy z rozszerzeniem .vhd!** Program automatycznie generuje plik **.vhd*. Dla wygody i zgodności nazw z projektem *./z2* nazwa pliku dla generowanej tablicy prawdy powinna mieć nazwę *da_table.pla*. W wyniku działania programu *filter.exe* otrzymamy dwa pliki: *da_table.pla* oraz *da_table.vhd*.

```
Administrator: C:\Windows\System32\cmd.exe
Microsoft Windows [Wersja 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Wszelkie prawa zastrzeżone.

h:\>filter.exe

podaj nazwe pliku ze wspolczynnikami: z2.txt
podaj nazwe pliku dla tablicy LUT: da_table.pla

h:\>_
```

Wydruk pliku *da_table.vhd*:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_arith.ALL;
USE ieee.numeric_std.ALL;
```

```

ENTITY da_table IS
    PORT ( da_i   : IN STD_LOGIC_VECTOR(2 DOWNTO 0);
          da_o   : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
        );
END da_table;

ARCHITECTURE LCs OF da_table IS
    signal table_out: std_LOGIC_VECTOR(7 DOWNTO 0);
BEGIN
    da_o <= table_out;

PROCESS (da_i)
BEGIN
    CASE da_i IS
        WHEN "000" => table_out <= "00000000";
        WHEN "001" => table_out <= "11100111";
        WHEN "010" => table_out <= "01001100";
        WHEN "011" => table_out <= "00110011";
        WHEN "100" => table_out <= "11100111";
        WHEN "101" => table_out <= "11001110";
        WHEN "110" => table_out <= "00110011";
        WHEN "111" => table_out <= "00011010";
        WHEN OTHERS => table_out <= "00000000";
    END CASE;
END PROCESS;

END LCs;

```

Zadanie 2b. umieścić wygenerowany plik *da_table.vhd* w katalogu *./z2*. Zwrócić uwagę na poprawne wartości parametrów w pliku *filtr_z2.vhd* określające wielkość tablicy DA.

Wydruk fragmentu pliku *filtr_z2.vhd*:

```

66
67 -- DA TABLE component declaration
68
69 component da_table is
70     port
71         da_i   : in  STD_LOGIC_VECTOR(2 downto 0);
72         da_o   : out STD_LOGIC_VECTOR(7 downto 0);
73
74     end component;
75
76 -- end of DA TABLE component declaration
77
78
79 signal da_in_s   : STD_LOGIC_VECTOR(2 downto 0);
80 signal da_out_s  : STD_LOGIC_VECTOR(7 downto 0);
81
82
83 begin
84
85
86 U_FILTER_DA_TOP:filter_da_top -- filter and codec machines inst
87
88 generic map(
89     TAPS_NO    => 3,
90     DAOUT_BITS => 8
91 )
92 port map(
93     CLOCK      => CLOCK,
94     RESET      => RESET,
95
96     -- codec interface
97
98     AUD_XCK    => AUD_XCK,

```

Skompilować projekt i zanotować zużycie zasobów logicznych układu fpga (komórek, pamięci, mnożarek) oraz maksymalną częstotliwość pracy f_{max} . Zaprogramować układ fpga. Włączyć w programie Specrum Lab generator szumu i zaobserwować na wykresie działanie filtru, odpowiednio zmieniając ustawienie przełącznika SW0.

2.4. Symetria i synchroniczne wyjście

- W zadaniu 1. wstawić na wyjściu filtru rejestr. Zaobserwować wyniki po kompilacji zużytych zasobów i f_{max} .
- Zmodyfikować zadanie 1. wykorzystując fakt że współczynniki filtru są symetryczne. Zaobserwować wyniki po kompilacji zużytych zasobów i f_{max} .

Literatura i materiały pomocnicze:

1. Plansze do wykładu UCYF
2. Literatura podana na wykładzie, ze szczególnym uwzględnieniem rozdz. 6 książki „Programowalne układy przetwarzania sygnałów i informacji”
3. Dokument PDF: „DE2-115 User Manual”
4. Program Spectrum Lab, <http://www.qsl.net/dl4yhf/spectra1.html>
5. Program SciLab, <http://www.scilab.org/>

*Opracowanie wewnętrzne ZCB IT, PW, styczeń 2016:
M. Staworko, P. Tomaszewicz*