FUNDAMENTAL OF DIGITAL SYSTEM FINAL PROJECT REPORT
DEPARTMENT OF ELECTRICAL ENGINEERING
UNIVERSITAS INDONESIA

LOW AND HIGH PASS FILTER

GROUP AP10

Muhammad Daffa Rizkyandri	**2206829194**
Wendy Dharmawan	**2206059591**
Muhammad Zhavier Naufal Rachman	**2206063001**
Reiki Putra Darmawan	**2206062882**

PREFACE

Laporan proyek akhir ini telah dipersiapkan dalam rangka pemenuhan sebagian persyaratan untuk program studi: Perancangan Sistem Digital dan Praktikum (Sem. III) dalam tahun akademik 2023 - 2024.

Dalam mempersiapkan laporan ini, kita telah membuat desain program VHDL yang telah kita rancang dalam waktu yang disediakan untuk membantu kami dalam perancangan proyek akhir ini dan untuk meginformasikan keterangan yang diperlukan. Perpaduan antara pembelajaran yang kami dapat dari kelas dan praktikum yang formal maupun informal akan direpresentasikan dalam laporan proyek akhir ini.

Proyek ini menggunakan modul 2, 3, 4, 5, 6, 7, 8, 9 dari praktikum perancangan sistem digital yang berisi behavioral style, testbench, structural style, looping construct, procedure, function, impure function, finite state machine, dan microprogramming untuk menyelesaikan desain program VHDL low and high pass filter. Inti dari ide proyek ini program berguna untuk melakukan filtering dari sebuah file audio, dengan melakukan cutoff terhadap frekuensi audio di rentang tertentu.

Depok, December 24, 2023

Group AP10

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

Dalam Digital Signal Processing (DSP), Digital Filter merupakan sistem yang digunakan untuk mengubah suatu sinyal *input* menjadi bentuk sinyal yang diinginkan sebagai *output*. Filter ini dapat digunakan untuk memisahkan sinyal atau memulihkan sinyal. Salah satu filter yang sering digunakan disebut *finite impulse response* (FIR) filter. Filter ini adalah filter yang respons impulsnya memiliki periode terbatas, karena dalam waktu terbatas ia akan menjadi nol. Dalam proyek ini, kami memutuskan untuk merancang filter FIR waktu diskrit kausal sederhana yang memungkinkan lewatnya frekuensi tinggi atau rendah, dan memodifikasi sinyal keluarannya.

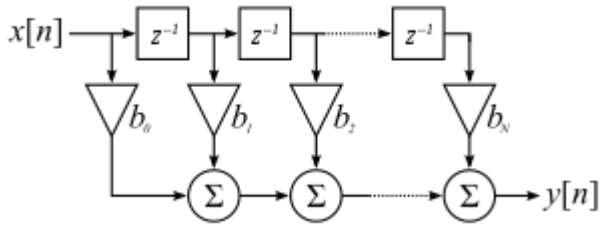
1.2 PROJECT DESCRIPTION

Seperti yang disebutkan sebelumnya projek yang dibuat adalah FIR filter. Filter ini merupakan tipe filter *feedforward*, yang artinya filter bekerja hanya berdasarkan input saat ini dan input sebelumnya, berbeda dengan feedback filter yang menggunakan output hasil sebelumnya.

Penerapan FIR filter dilakukan menggunakan diskrit kausal rendah, yang memiliki fungsi output sebagai berikut:

$$\begin{aligned} y[n] &= b_0 x[n] + b_1 x[n-1] + \cdots + b_N x[n-N] \\ &= \sum_{i=0}^{N} b_i \cdot x[n-i], \end{aligned}$$

Perumusan tersebut dapat diterapkan dalam bentuk rangkaian atau bentuk diskrit seperti gambar berikut:



Dengan X[n] sebagai input biner yang dimasukkan, Z sebagai sample-delay berorder, b sebagai amplifier dan sigma sebagai adder yang akan menghasilkan output Y[n] yaitu hasil filter.

Dengan mengatur jumlah order, dan koefisien dari masing-masing amplifier, maka program ini dapat digunakan untuk membentuk berbagai jenis filter. Contohnya dengan menerapkan order sebanyak 1 dengan koefisien amplifier sebesar 0.5 di kedua amplifier, akan terbentuk low pass filter yang meneruskan bagian frequensi rendah dari sinyal.

1.3 OBJECTIVES

Tujuan dari proyek ini adalah sebagai berikut:

1. Membuat rangkaian controller utama untuk FIR Filter
2. Membuat komponen-komponen untuk filter
3. Membuat sample delay
4. Membuat testbench untuk menjalankan filter

1.4 ROLES AND RESPONSIBILITIES

The roles and responsibilities assigned to the group members are as follows:

Roles	Responsibilities	Person
Role 1	Pembuatan program utama, pembuatan laporan	Wendy Dharmawan
Role 2	Pembuatan Testbench, pembuatan laporan	M Daffa Rizkyandri
Role 3	Pembuatan PPT	Muhammad Zhavier
Role 4	Pembutan PPT	Reiki Putra

Table 1. Roles and Responsibilities

CHAPTER 2

IMPLEMENTATION

2.1 EQUIPMENT

The tools that are going to be used in this project are as follows:

- Visual Studio Code text editor software
- ModelSim HDL simulator
- Intel Quartus Prime FPGA design software
- Python untuk menampilkan grafik dengan matplotlib

2.2 IMPLEMENTATION

Dalam pembuatan program ini diimplementasikan modul 2 hingga 9, detail dari pengimplementasian modul adalah sebagai berikut:

- Data Flow Style

Data flow style diterapkan dalam component adder, dimana output berada pada gate-level dan di implementasikan secara conccurent.

```
a_out <= (real_convert(adder_in(0)) + real_convert(adder_in(1))
+ real_convert(adder_in(2)) + real_convert(adder_in(3))
+ real_convert(adder_in(4)) + real_convert(adder_in(5))
+ real_convert(adder_in(6)) + real_convert(adder_in(7)));
```

- Behavioural Style

Behavioural style diterapkan dalam bagian program yang menerapkan looping. Dalam bagian program ini, digunakan process sehingga program dapat berjalan secara sekuensial.

```
when COUNT =>
    if rising_edge(clk) then
        for i in 0 to 7 loop
            s_amp_in(i) <= s_delay_out(i);
        end loop;
    end if;
    state <= ADD;
```

- Testbench

Testbench digunakan untuk mensimulasikan dan menjalankan program, pada testbench memanfaatkan port mapping.

FIR_Filter_tb.vhd

```vhdl
-- FILEPATH: /d:/Wendy/PSD/FIR_Filter_tb.vhd
library ieee;
use ieee.std_logic_1164.all;

entity FIR_Filter_tb is
end FIR_Filter_tb;

architecture tb_arch of FIR_Filter_tb is
    -- Component declaration
    component FIR_Filter is
        port (
            clk : in std_logic;
            control_word : in std_logic_vector(7 downto 0)
        );
    end component FIR_Filter;

    -- Signal declaration
    signal clk_tb : std_logic := '0';
    signal control_word_tb : std_logic_vector(7 downto 0) := (others => '0');
```

- Structural Programming

Structural programming adalah metode programming pada VHDL menggunakan component dan port mapping. Metode programming ini diterapkan pada bagian controller filter.

```vhdl
component Adder is
    port (
        adder_in : in array_8; -- Input to the adder component
        adder_out : out std_logic_vector(7 downto 0) -- Output from the adder component
    );
end component Adder;

component Amplifier is
    port(
        coefficient : in std_logic_vector(7 downto 0); -- Coefficient input to the amplifier componen
        amp_in : in std_logic_vector(7 downto 0); -- Input to the amplifier component
        amp_out : out std_logic_vector(7 downto 0) -- Output from the amplifier component
    );
end component Amplifier;
```

- Looping Construct

Digunakan pada bagian bagian kode behavioral, dan bagian for generate untuk menginstantiasi banyak komponen dalam satu loop.

```
amp_delay: for i in 0 to 7 generate
    gen_delay: Sample_Delay port map(
        data_current => data_c, -- Current data index
        order => i, -- Order of the filter
        delay_in => s_delay_in, -- Delay input signal
        delay_out => s_delay_out(i) -- Delay output signal
    );
    gen_amp: Amplifier port map(
        coefficient => s_coefficient_list(i), -- Coefficient input signal
        amp_in => s_amp_in(i), -- Amplifier input signal
        amp_out => s_amp_out(i) -- Amplifier output signal
    );
end generate;
```

- Function

Function yang digunakan dalam program ini adalah function untuk mengconvert bentuk std_logic_vector menjadi bentuk real untuk merepresentasikan angka floating point antara -1 dan 1. Function digunakan untuk bagian kode berulang.

```
function real_convert (x : std_logic_vector) return real is
    variable temp : real;
begin
    temp := real(to_integer(signed(x)));
    return temp;
end function real_convert;
```

- FSM

Pada program terdapat beberapa state, yaitu FETCH_ORDER, FETCH_INPUT, FETCH_AMP, COUNT, ADD, DONE. Pada state-state ini jenis input dan operasi yang dilakukan berbeda-beda.

```
if rising_edge(clk) then
    case state is
        when FETCH_ORDER=>
            order <= to_integer(unsigned(control_word(7 downto 5))); --
            state <= FETCH_INPUT;

        when FETCH_INPUT=>
            if rising_edge(clk) then
                s_delay_in(data_c) <= control_word(7 downto 0); -- Store
                data_c <= data_c + 1; -- Increment data index
            end if;
            if data_c = order then
                data_c <= 0;
                state <= FETCH_AMP;
            end if;

        when FETCH_AMP =>
```

- Microprogramming

Pada program digunakan control word sebagai input. Control word digunakan dalam menentukan jumlah order dan dikonversikan menjadi nilai koefisien dan input sinyal.

```
order <= to_integer(unsigned(control_word(7 downto 5)));
```

```
s_delay_in(data_c) <= control_word(7 downto 0);
```

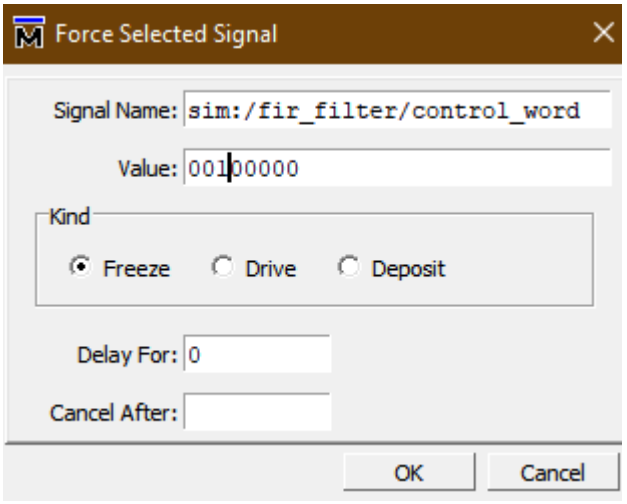
CHAPTER 3

TESTING AND ANALYSIS

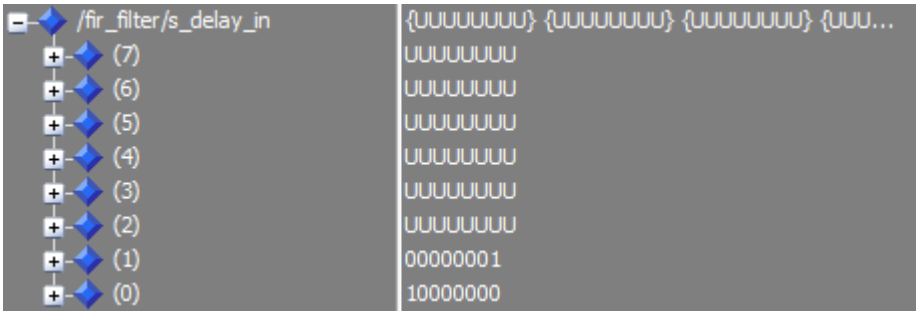
3.1 TESTING

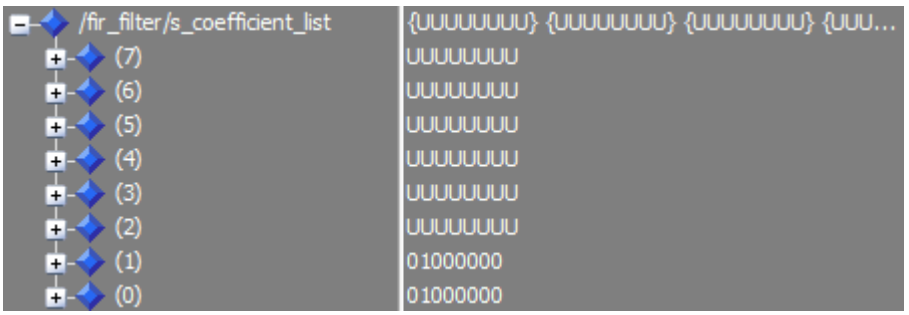
Pada bagian testing, program akan mencoba mensimulasikan sebuah low pass filter, yaitu filter dengan order sebanyak 1 (order 0 dan order 1) dan nilai masing-masing koefisien amplifier sebanyak 0.5 (64/128).

Untuk input control word dibagian fetch order, cukup mengisi 3 bit pertama untuk menentukan jumlah order.



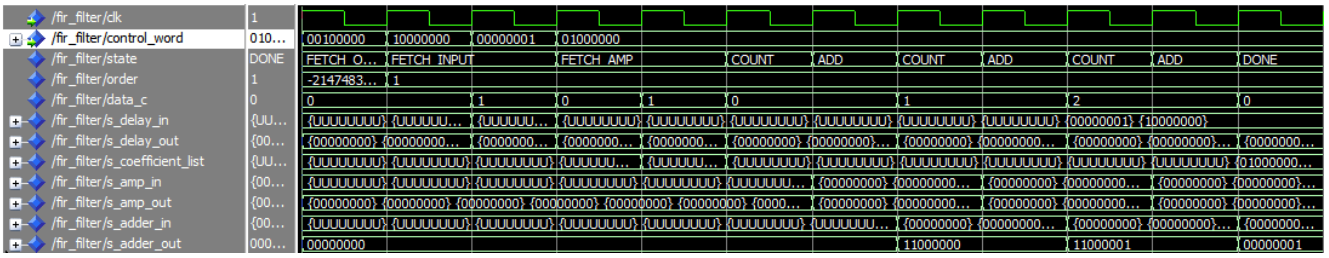
Untuk input sinyal diisi pada state fetch input secara bergantian. Pada demonstrasi kali ini digunakan input 10000000, 00000001 dan input coefficient amplifier 01000000 (64).

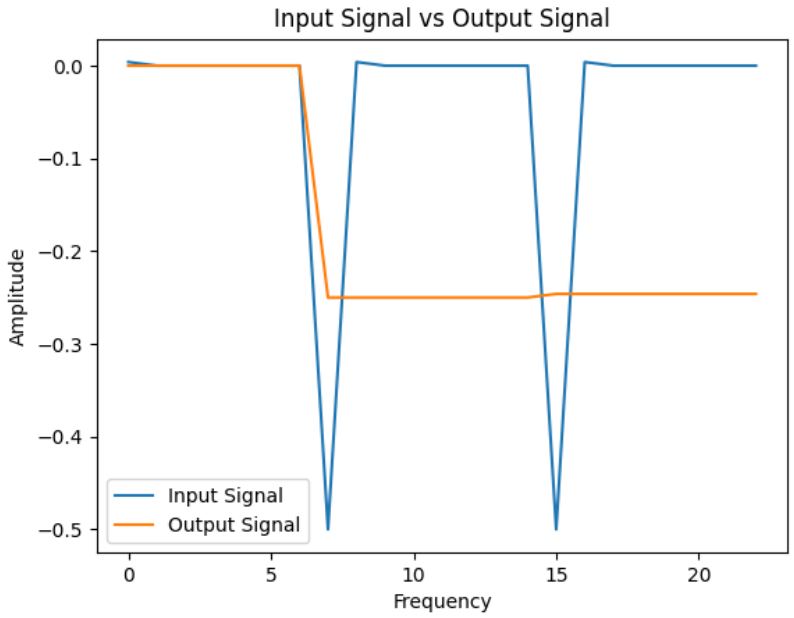




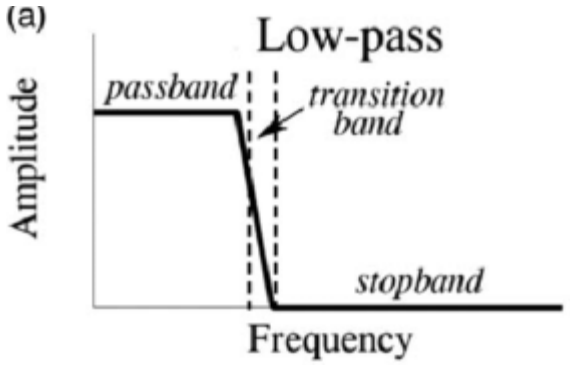
Setelah itu, program dijalankan hingga menyelesaikan perhitungan dan memasuki state done. Hasil grafik program dapat dilihat dengan menjalankan program python

3.2 RESULT





Dari hasil dapat terlihat bahwa program berhasil menghasilkan sebuah lowpass filter, yang hanya meneruskan amplitudo dari bagian frekuensi rendah.



3.3 ANALYSIS

Dari hasil yang dijalankan, dapat dilihat bahwa program sudah dapat membentuk filter untuk sinyal berdasarkan input yang diberikan pada control word. Program ini bekerja dengan berprinsip pada diskrit kausal rendah melalui komponen sample-delay, amplifier, dan adder.

Untuk menerapkan filter yang diinginkan, pengguna harus mengetahui jumlah order dan koefisien yang diperlukan. Nilai-nilai ini kemudian di apply kepada input sinyal pada bagian komponen.

Terdapat beberapa kekurangan dari program ini. Karena program ini menggunakan real number untuk merepresentasikan bilangan floating point dari -1 ke 1 sebagai koefisien, maka

program ini tidak dapat disintesis. Selain itu, proses konversi antar std_logic_vector real number dapat mengurangi presisi dari koefisien sehingga hasil masih belum sepenuhnya akurat. Pada result, dapat dilihat bahwa nilai amplitudo belum sesuai dengan harapan, yaitu amplitudo berada di atas 0 dan di cutoff menjadi 0. Hal ini diduga karena input sinyal berupa std_logic_vector sehingga belum sempurna. Namun secara general, program ini sudah berhasil untuk menerapkan filtering pada input sinyal dengan prinsip diskrit kausal rendah.

CHAPTER 4

CONCLUSION

Program FIR Filter yang dibuat sudah dapat berjalan dan mengimplementasikan filter pada input sinyal berupa std_logic_vector. Meskipun terdapat beberapa ketidakpresisian dan kesalahan pada amplitudo, program ini dapat menerapkan filtering sinyal melalui diskrit kausal rendah menggunakan prinsip-prinsip yang dipelajari pada Perancangan Sistem Digital, diantaranya menggunakan konsep state machine, microprogramming, component dan structural programming.

REFERENCES

[1] "Finite impulse response," *Wikipedia*, Sep. 17, 2021. https://en.wikipedia.org/wiki/Finite_impulse_response

[2] "Digital Filter Basics - YouTube," *www.youtube.com*. https://youtube.com/playlist?list=PLbqhA-NKGP6Afr_KbPUuy_yIBpPR4jzWo&si=y9aBnQ-xpck41-Ll (accessed Dec. 24, 2023).

[3] Surf-VHDL, "How to Implement FIR Filter in VHDL," *Surf-VHDL*, Nov. 14, 2015. https://surf-vhdl.com/how-to-implement-fir-filter-in-vhdl/ (accessed Dec. 24, 2023).

[4] "4.3: Discrete Time Convolution," *Engineering LibreTexts*, May 23, 2020. https://eng.libretexts.org/Bookshelves/Electrical_Engineering/Signal_Processing_and_Modeling/Signals_and_Systems_(Baraniuk_et_al.)/04%3A_Time_Domain_Analysis_of_Discrete_Time_Systems/4.03%3A_Discrete_Time_Convolution (accessed Sep. 09, 2023).

[5] R. Kumar, 2Rishabh Kumar, S. Ahmed Dar, and Mahantesh S Patil, *An Analysis on FIR Filter-Efficiency (FFE)*.

APPENDICES

Appendix A: Project Schematic

Kode tidak dapat disintesis di Quartus, dikarenakan menggunakan real number.

Appendix B: Documentation

Finite State Machine: