TUGAS AKHIR MATA KULIAH PERANCANGAN KOMPONEN TERPROGRAM : HIGH PASS FILTER DIGITAL FIR MENGGUNAKAN FPGA DE10-LITE

Oleh :

M. Taufiqul Huda 5022211007

Dosen Pengampu :

Dr. Rudy Dikairono, ST., MT.

DEPARTEMEN TEKNIK ELEKTRO

FAKULTAS TEKNOLOGI ELEKTRO DAN INFORMATIKA CERDAS

INSTITUT TEKNOLOGI SEPULUH NOPEMBER

SURABAYA

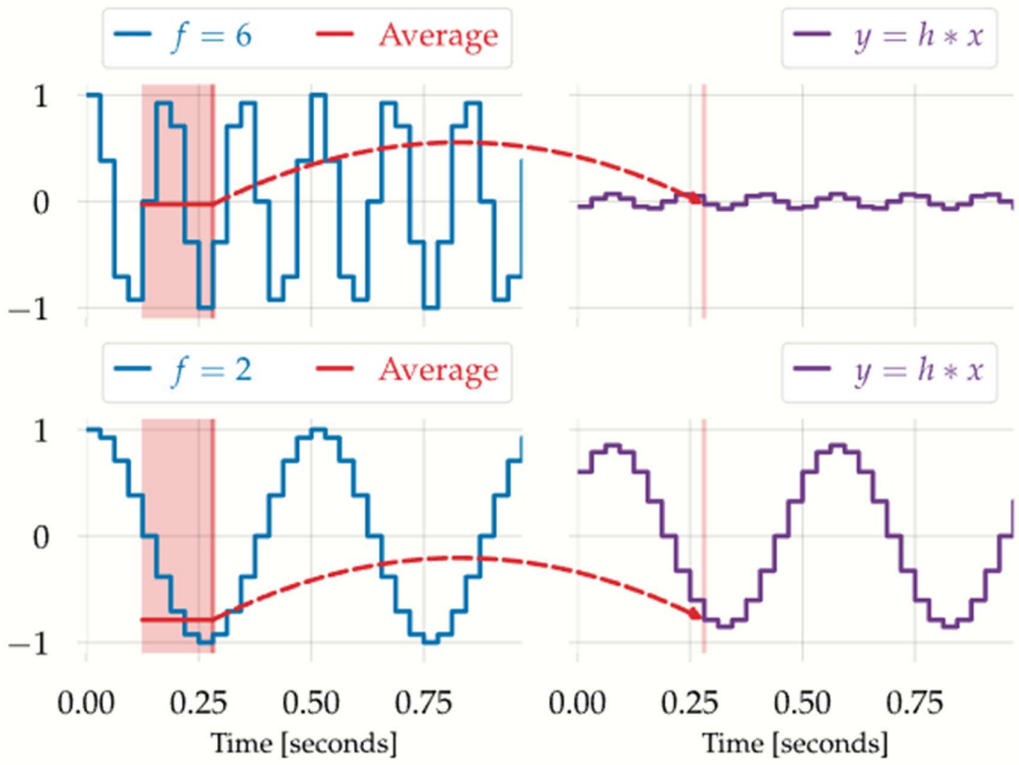
2023

Daftar Isi

Daftar Isi……2
A. Pengantar Teori Dasar Filter dan FPGA....3
B. Program VHDL Filter Digital FIR (*High Pass Filter*)....4
1) Modul Pembagi Frekuensi....8
2) Modul Filter HPF Digital FIR....9
C. Implementasi Pada FPGA dan Hasil Uji Coba....22
1) Hasil Uji Coba....23
D. Kesimpulan....25

A. Pengantar Teori Dasar Filter dan FPGA

Filter digital adalah proses mefilter sinyal berdasarkan respon frekuensi. Jenis filter sangat beragam berdasarkan karakternya, ada filter low pass, high pass, band pass, dan band stop. Pada proyek ini akan digunakan jenis filter high pass dengan orde filter sebesar -40 dB/decade atau orde 3. Prinsip dari filter digital FIR adalah mengkonvolusi sinyal input dengan kernel tertentu yang tergantung jenis filternya. Ilustrasi operasi konvolusi dapat dilihat pada gambar 1.



Gambar 1. Ilustrasi operasi konvolusi filter LPF digital dengan FIR

Sehingga untuk merencanakan sebuah filter digital FIR dengan frekuensi *cut-off* sebesar 1900 Hz (500 + 200*(NRP mod 10)), perlu merencanakan ukuran kernel dan konstanta kernel. Berikut adalah algoritma dasar program filter digital FIR.

```
#include "SampleFilter.h"

static int filter_taps[SAMPLEFILTER_TAP_NUM] = {
  -85,
  131,
  22,
  -58,
  -83,
  -20,
  95,
  147,
  24,
  -266,
  -574,
  1341,
  -574,
```

```
 -266,
 24,
 147,
 95,
 -20,
 -83,
 -58,
 22,
 131,
 -85
};

void SampleFilter_init(SampleFilter* f) {
 int i;
 for(i = 0; i < SAMPLEFILTER_TAP_NUM; ++i)
  f->history[i] = 0;
 f->last  index = 0;
}

void SampleFilter_put(SampleFilter* f, int input) {
 f->history[f->last_index++] = input;
 if(f->last_index == SAMPLEFILTER_TAP_NUM)
  f->last_index = 0;
}

int SampleFilter_get(SampleFilter* f) {
 long long acc = 0;
 int index = f->last_index, i;
 for(i = 0; i < SAMPLEFILTER_TAP_NUM; ++i) {
  index = index != 0 ? index-1 : SAMPLEFILTER  TAP  NUM-1;
  acc += (long long)f->history[index] * filter_taps[i];
 };
 return acc >> 12;
}
```

Program di atas melakukan proses konvolusi nilai input dengan kernel sehingga sinyal input dapt terfilter dengan sesuai rencana filter yang di representasikan dalam bentuk kernel. Dari algoritma tersebut akan di implementasikan dalam program VHDL sehingga dapat di implementasikan dalam FPGA.

B. Program VHDL Filter Digital FIR (*High Pass Filter*)

Pada rencana program VHDL diperlukan 3 modul entity VHDL, yaitu : *ADC Intelectual Proprties libraries entity*, *Clock divider*, dan modul filter itu sendiri. Modul ADC memiliki resolusi sebesar 12 bit, maka dalam rencana filter akan dikeluarkan hasil filter dengan resolusi setidaknya sama yaitu 12 bit. Modul pembagi clock berfungsi untuk membagi clock osilator FPGA DE10-Lite yang awalnya 10 MHz (Clock ADC FPGA Intel Altera DE10-Lite) menjadi 20 kHz (sesuai rencana filter dengan frekuensi sampling 20 kHz). Sehingga bandwidth dari filter adalah setengah dari frekuensi sampling yaitu 0 Hz – 10 kHz. Modul filter digital *high pass filter* sejatinya adalah modul untuk melakukan operasi konvolusi terhadap input menggunakan kernel yang direncanakan. Berikut kernel yang digunakan dalam operasi konvolusi filter FIR.

```
Kernel [155] = (
            x"00B",
            x"FE0",
            x"007",
            x"009",
```

x"000",
x"FFA",
x"FFB",
x"001",
x"006",
x"004",
x"FFD",
x"FFA",
x"FFE",
x"004",
x"006",
x"000",
x"FFA",
x"FFB",
x"002",
x"007",
x"004",
x"FFB",
x"FF8",
x"FFF",
x"007",
x"007",
x"FFE",
x"FF7",
x"FFA",
x"005",
x"00A",
x"003",
x"FF7",
x"FF6",
x"001",
x"00C",
x"008",
x"FFA",
x"FF2",
x"FFB",
x"00B",
x"00E",
x"000",
x"FF1",
x"FF4",
x"006",
x"012",
x"008",
x"FF3",
x"FEC",

```
x"FFE",
x"014",
x"013",
x"FF9",
x"FE6",
x"FF2",
x"011",
x"01E",
x"006",
x"FE3",
x"FE1",
x"007",
x"029",
x"01A",
x"FE7",
x"FCB",
x"FF1",
x"032",
x"03F",
x"FF9",
x"FA8",
x"FB9",
x"039",
x"0A2",
x"04C",
x"F14",
x"DA9",
x"508",
x"DA9",
x"F14",
x"04C",
x"0A2",
x"039",
x"FB9",
x"FA8",
x"FF9",
x"03F",
x"032",
x"FF1",
x"FCB",
x"FE7",
x"01A",
x"029",
x"007",
x"FE1",
x"FE3",
```

x"006",
x"01E",
x"011",
x"FF2",
x"FE6",
x"FF9",
x"013",
x"014",
x"FFE",
x"FEC",
x"FF3",
x"008",
x"012",
x"006",
x"FF4",
x"FF1",
x"000",
x"00E",
x"00B",
x"FFB",
x"FF2",
x"FFA",
x"008",
x"00C",
x"001",
x"FF6",
x"FF7",
x"003",
x"00A",
x"005",
x"FFA",
x"FF7",
x"FFE",
x"007",
x"007",
x"FFF",
x"FF8",
x"FFB",
x"004",
x"007",
x"002",
x"FFB",
x"FFA",
x"000",
x"006",
x"004",

```
        x"FFE",
        x"FFA",
        x"FFD",
        x"004",
        x"006",
        x"001",
        x"FFB",
        x"FFA",
        x"000",
        x"009",
        x"007",
        x"FE0",
        x"00B");
```

Sehingga dari kernel tersebut dapat dibuat filter *high pass filter* digital FIR menggunakan konvolusi. Berikut program VHDL tiap modul (pembagi frekuensi dan filter).

1) Modul Pembagi Frekuensi

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
USE IEEE.std_logic_signed.all;
use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity clk_100div is
   Port ( CLK : in STD_LOGIC;
          CLK_OUT : out STD_LOGIC);
end clk_100div;

architecture Behavioral of clk_100div is
   signal count : INTEGER range 0 to 1000 := 0;
            signal clk_buf : STD_LOGIC := '0';

begin
   process(CLK)
   begin
      if rising_edge(CLK) then
         if (count = 500) then
            clk_buf <= not clk_buf;
            count <= 0; -- Reset count to 0
         else
            count <= count + 1;
         end if;
```

```
      end if;
   end process;
   CLK_OUT <= clk_buf;
end Behavioral;
```

2) Modul Filter HPF Digital FIR

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
USE IEEE.std_logic_signed.all;
use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity filter is
   Port ( EN : in STD_LOGIC;
          CLK : in STD_LOGIC;
          INPUT_ADC : in STD_LOGIC_VECTOR (11 downto 0);
          OUTPUT_FILTERED : out STD_LOGIC_VECTOR (11 downto 0));
end filter;

architecture Behavioral of filter is
   type array_type is array (0 to 154) of std_logic_vector(11 downto 0);
   signal i : INTEGER;
   signal sum : std_logic_vector(23 downto 0) := (others => '0');
   signal new_input : std_logic_vector (11 downto 0);
          signal x_sample : array_type := (others => x"000");
   signal coeff : array_type := (
                     x"00B",
                     x"FE0",
                     x"007",
                     x"009",
                     x"000",
                     x"FFA",
                     x"FFB",
                     x"001",
                     x"006",
                     x"004",
                     x"FFD",
                     x"FFA",
                     x"FFE",
                     x"004",
                     x"006",
                     x"000",
```

```
x"FFA",
x"FFB",
x"002",
x"007",
x"004",
x"FFB",
x"FF8",
x"FFF",
x"007",
x"007",
x"FFE",
x"FF7",
x"FFA",
x"005",
x"00A",
x"003",
x"FF7",
x"FF6",
x"001",
x"00C",
x"008",
x"FFA",
x"FF2",
x"FFB",
x"00B",
x"00E",
x"000",
x"FF1",
x"FF4",
x"006",
x"012",
x"008",
x"FF3",
x"FEC",
x"FFE",
x"014",
x"013",
x"FF9",
x"FE6",
x"FF2",
x"011",
x"01E",
x"006",
x"FE3",
x"FE1",
x"007",
x"029",
x"01A",
x"FE7",
x"FCB",
x"FF1",
```

x"032",
x"03F",
x"FF9",
x"FA8",
x"FB9",
x"039",
x"0A2",
x"04C",
x"F14",
x"DA9",
x"508",
x"DA9",
x"F14",
x"04C",
x"0A2",
x"039",
x"FB9",
x"FA8",
x"FF9",
x"03F",
x"032",
x"FF1",
x"FCB",
x"FE7",
x"01A",
x"029",
x"007",
x"FE1",
x"FE3",
x"006",
x"01E",
x"011",
x"FF2",
x"FE6",
x"FF9",
x"013",
x"014",
x"FFE",
x"FEC",
x"FF3",
x"008",
x"012",
x"006",
x"FF4",
x"FF1",
x"000",
x"00E",
x"00B",
x"FFB",
x"FF2",
x"FFA",

```
                    x"008",
                    x"00C",
                    x"001",
                    x"FF6",
                    x"FF7",
                    x"003",
                    x"00A",
                    x"005",
                    x"FFA",
                    x"FF7",
                    x"FFE",
                    x"007",
                    x"007",
                    x"FFF",
                    x"FF8",
                    x"FFB",
                    x"004",
                    x"007",
                    x"002",
                    x"FFB",
                    x"FFA",
                    x"000",
                    x"006",
                    x"004",
                    x"FFE",
                    x"FFA",
                    x"FFD",
                    x"004",
                    x"006",
                    x"001",
                    x"FFB",
                    x"FFA",
                    x"000",
                    x"009",
                    x"007",
                    x"FE0",
                    x"00B",
                    others => x"000"
); -- initialize with your coeff values

begin
   process(CLK)
   begin
      if rising_edge(CLK) and EN = '1' then -- added EN condition
                                    x_sample(154) <= x_sample(153);
                                    x_sample(153) <= x_sample(152);
                                    x_sample(152) <= x_sample(151);
                                    x_sample(151) <= x_sample(150);
                                    x_sample(150) <= x_sample(149);
                                    x_sample(149) <= x_sample(148);
                                    x_sample(148) <= x_sample(147);
```

```
x_sample(147) <= x_sample(146);
x_sample(146) <= x_sample(145);
x_sample(145) <= x_sample(144);
x_sample(144) <= x_sample(143);
x_sample(143) <= x_sample(142);
x_sample(142) <= x_sample(141);
x_sample(141) <= x_sample(140);
x_sample(140) <= x_sample(139);
x_sample(139) <= x_sample(138);
x_sample(138) <= x_sample(137);
x_sample(137) <= x_sample(136);
x_sample(136) <= x_sample(135);
x_sample(135) <= x_sample(134);
x_sample(134) <= x_sample(133);
x_sample(133) <= x_sample(132);
x_sample(132) <= x_sample(131);
x_sample(131) <= x_sample(130);
x_sample(130) <= x_sample(129);
x_sample(129) <= x_sample(128);
x_sample(128) <= x_sample(127);
x_sample(127) <= x_sample(126);
x_sample(126) <= x_sample(125);
x_sample(125) <= x_sample(124);
x_sample(124) <= x_sample(123);
x_sample(123) <= x_sample(122);
x_sample(122) <= x_sample(121);
x_sample(121) <= x_sample(120);
x_sample(120) <= x_sample(119);
x_sample(119) <= x_sample(118);
x_sample(118) <= x_sample(117);
x_sample(117) <= x_sample(116);
x_sample(116) <= x_sample(115);
x_sample(115) <= x_sample(114);
x_sample(114) <= x_sample(113);
x_sample(113) <= x_sample(112);
x_sample(112) <= x_sample(111);
x_sample(111) <= x_sample(110);
x_sample(110) <= x_sample(109);
x_sample(109) <= x_sample(108);
x_sample(108) <= x_sample(107);
x_sample(107) <= x_sample(106);
x_sample(106) <= x_sample(105);
x_sample(105) <= x_sample(104);
x_sample(104) <= x_sample(103);
x_sample(103) <= x_sample(102);
x_sample(102) <= x_sample(101);
x_sample(101) <= x_sample(100);
x_sample(100) <= x_sample(99);
x_sample(99) <= x_sample(98);
x_sample(98) <= x_sample(97);
x_sample(97) <= x_sample(96);
```

```
x_sample(96) <= x_sample(95);
x_sample(95) <= x_sample(94);
x_sample(94) <= x_sample(93);
x_sample(93) <= x_sample(92);
x_sample(92) <= x_sample(91);
x_sample(91) <= x_sample(90);
x_sample(90) <= x_sample(89);
x_sample(89) <= x_sample(88);
x_sample(88) <= x_sample(87);
x_sample(87) <= x_sample(86);
x_sample(86) <= x_sample(85);
x_sample(85) <= x_sample(84);
x_sample(84) <= x_sample(83);
x_sample(83) <= x_sample(82);
x_sample(82) <= x_sample(81);
x_sample(81) <= x_sample(80);
x_sample(80) <= x_sample(79);
x_sample(79) <= x_sample(78);
x_sample(78) <= x_sample(77);
x_sample(77) <= x_sample(76);
x_sample(76) <= x_sample(75);
x_sample(75) <= x_sample(74);
x_sample(74) <= x_sample(73);
x_sample(73) <= x_sample(72);
x_sample(72) <= x_sample(71);
x_sample(71) <= x_sample(70);
x_sample(70) <= x_sample(69);
x_sample(69) <= x_sample(68);
x_sample(68) <= x_sample(67);
x_sample(67) <= x_sample(66);
x_sample(66) <= x_sample(65);
x_sample(65) <= x_sample(64);
x_sample(64) <= x_sample(63);
x_sample(63) <= x_sample(62);
x_sample(62) <= x_sample(61);
x_sample(61) <= x_sample(60);
x_sample(60) <= x_sample(59);
x_sample(59) <= x_sample(58);
x_sample(58) <= x_sample(57);
x_sample(57) <= x_sample(56);
x_sample(56) <= x_sample(55);
x_sample(55) <= x_sample(54);
x_sample(54) <= x_sample(53);
x_sample(53) <= x_sample(52);
x_sample(52) <= x_sample(51);
x_sample(51) <= x_sample(50);
x_sample(50) <= x_sample(49);
x_sample(49) <= x_sample(48);
x_sample(48) <= x_sample(47);
x_sample(47) <= x_sample(46);
x_sample(46) <= x_sample(45);
```

```
                    x_sample(45) <= x_sample(44);
                    x_sample(44) <= x_sample(43);
                    x_sample(43) <= x_sample(42);
                    x_sample(42) <= x_sample(41);
                    x_sample(41) <= x_sample(40);
                    x_sample(40) <= x_sample(39);
                    x_sample(39) <= x_sample(38);
                    x_sample(38) <= x_sample(37);
                    x_sample(37) <= x_sample(36);
                    x_sample(36) <= x_sample(35);
                    x_sample(35) <= x_sample(34);
                    x_sample(34) <= x_sample(33);
                    x_sample(33) <= x_sample(32);
                    x_sample(32) <= x_sample(31);
                    x_sample(31) <= x_sample(30);
                    x_sample(30) <= x_sample(29);
                    x_sample(29) <= x_sample(28);
                    x_sample(28) <= x_sample(27);
                    x_sample(27) <= x_sample(26);
                    x_sample(26) <= x_sample(25);
                    x_sample(25) <= x_sample(24);
                    x_sample(24) <= x_sample(23);
                    x_sample(23) <= x_sample(22);
                    x_sample(22) <= x_sample(21);
                    x_sample(21) <= x_sample(20);
                    x_sample(20) <= x_sample(19);
                    x_sample(19) <= x_sample(18);
                    x_sample(18) <= x_sample(17);
                    x_sample(17) <= x_sample(16);
                    x_sample(16) <= x_sample(15);
                    x_sample(15) <= x_sample(14);
                    x_sample(14) <= x_sample(13);
                    x_sample(13) <= x_sample(12);
                    x_sample(12) <= x_sample(11);
                    x_sample(11) <= x_sample(10);
                    x_sample(10) <= x_sample(9);
                    x_sample(9) <= x_sample(8);
                    x_sample(8) <= x_sample(7);
                    x_sample(7) <= x_sample(6);
                    x_sample(6) <= x_sample(5);
                    x_sample(5) <= x_sample(4);
                    x_sample(4) <= x_sample(3);
                    x_sample(3) <= x_sample(2);
                    x_sample(2) <= x_sample(1);
                    x_sample(1) <= x_sample(0);
x_sample(0) <= (INPUT_ADC);

                    sum <= std_logic_vector(

 signed(x_sample(154)) * signed(coeff(154)) +
```

```
signed(x_sample(153)) * signed(coeff(153)) +
signed(x_sample(152)) * signed(coeff(152)) +
signed(x_sample(151)) * signed(coeff(151)) +
signed(x_sample(150)) * signed(coeff(150)) +
signed(x_sample(149)) * signed(coeff(149)) +
signed(x_sample(148)) * signed(coeff(148)) +
signed(x_sample(147)) * signed(coeff(147)) +
signed(x_sample(146)) * signed(coeff(146)) +
signed(x_sample(145)) * signed(coeff(145)) +
signed(x_sample(144)) * signed(coeff(144)) +
signed(x_sample(143)) * signed(coeff(143)) +
signed(x_sample(142)) * signed(coeff(142)) +
signed(x_sample(141)) * signed(coeff(141)) +
signed(x_sample(140)) * signed(coeff(140)) +
signed(x_sample(139)) * signed(coeff(139)) +
signed(x_sample(138)) * signed(coeff(138)) +
signed(x_sample(137)) * signed(coeff(137)) +
signed(x_sample(136)) * signed(coeff(136)) +
signed(x_sample(135)) * signed(coeff(135)) +
signed(x_sample(134)) * signed(coeff(134)) +
signed(x_sample(133)) * signed(coeff(133)) +
signed(x_sample(132)) * signed(coeff(132)) +
signed(x_sample(131)) * signed(coeff(131)) +
signed(x_sample(130)) * signed(coeff(130)) +
signed(x_sample(129)) * signed(coeff(129)) +
```

```
signed(x_sample(128)) * signed(coeff(128)) +
signed(x_sample(127)) * signed(coeff(127)) +
signed(x_sample(126)) * signed(coeff(126)) +
signed(x_sample(125)) * signed(coeff(125)) +
signed(x_sample(124)) * signed(coeff(124)) +
signed(x_sample(123)) * signed(coeff(123)) +
signed(x_sample(122)) * signed(coeff(122)) +
signed(x_sample(121)) * signed(coeff(121)) +
signed(x_sample(120)) * signed(coeff(120)) +
signed(x_sample(119)) * signed(coeff(119)) +
signed(x_sample(118)) * signed(coeff(118)) +
signed(x_sample(117)) * signed(coeff(117)) +
signed(x_sample(116)) * signed(coeff(116)) +
signed(x_sample(115)) * signed(coeff(115)) +
signed(x_sample(114)) * signed(coeff(114)) +
signed(x_sample(113)) * signed(coeff(113)) +
signed(x_sample(112)) * signed(coeff(112)) +
signed(x_sample(111)) * signed(coeff(111)) +
signed(x_sample(110)) * signed(coeff(110)) +
signed(x_sample(109)) * signed(coeff(109)) +
signed(x_sample(108)) * signed(coeff(108)) +
signed(x_sample(107)) * signed(coeff(107)) +
signed(x_sample(106)) * signed(coeff(106)) +
signed(x_sample(105)) * signed(coeff(105)) +
signed(x_sample(104)) * signed(coeff(104)) +
```

```
signed(x_sample(103)) * signed(coeff(103)) +
signed(x_sample(102)) * signed(coeff(102)) +
signed(x_sample(101)) * signed(coeff(101)) +
signed(x_sample(100)) * signed(coeff(100)) +
signed(x_sample(99)) * signed(coeff(99)) +
signed(x_sample(98)) * signed(coeff(98)) +
signed(x_sample(97)) * signed(coeff(97)) +
signed(x_sample(96)) * signed(coeff(96)) +
signed(x_sample(95)) * signed(coeff(95)) +
signed(x_sample(94)) * signed(coeff(94)) +
signed(x_sample(93)) * signed(coeff(93)) +
signed(x_sample(92)) * signed(coeff(92)) +
signed(x_sample(91)) * signed(coeff(91)) +
signed(x_sample(90)) * signed(coeff(90)) +
signed(x_sample(89)) * signed(coeff(89)) +
signed(x_sample(88)) * signed(coeff(88)) +
signed(x_sample(87)) * signed(coeff(87)) +
signed(x_sample(86)) * signed(coeff(86)) +
signed(x_sample(85)) * signed(coeff(85)) +
signed(x_sample(84)) * signed(coeff(84)) +
signed(x_sample(83)) * signed(coeff(83)) +
signed(x_sample(82)) * signed(coeff(82)) +
signed(x_sample(81)) * signed(coeff(81)) +
signed(x_sample(80)) * signed(coeff(80)) +
signed(x_sample(79)) * signed(coeff(79)) +
```

```
signed(x_sample(78)) * signed(coeff(78)) +
signed(x_sample(77)) * signed(coeff(77)) +
signed(x_sample(76)) * signed(coeff(76)) +
signed(x_sample(75)) * signed(coeff(75)) +
signed(x_sample(74)) * signed(coeff(74)) +
signed(x_sample(73)) * signed(coeff(73)) +
signed(x_sample(72)) * signed(coeff(72)) +
signed(x_sample(71)) * signed(coeff(71)) +
signed(x_sample(70)) * signed(coeff(70)) +
signed(x_sample(69)) * signed(coeff(69)) +
signed(x_sample(68)) * signed(coeff(68)) +
signed(x_sample(67)) * signed(coeff(67)) +
signed(x_sample(66)) * signed(coeff(66)) +
signed(x_sample(65)) * signed(coeff(65)) +
signed(x_sample(64)) * signed(coeff(64)) +
signed(x_sample(63)) * signed(coeff(63)) +
signed(x_sample(62)) * signed(coeff(62)) +
signed(x_sample(61)) * signed(coeff(61)) +
signed(x_sample(60)) * signed(coeff(60)) +
signed(x_sample(59)) * signed(coeff(59)) +
signed(x_sample(58)) * signed(coeff(58)) +
signed(x_sample(57)) * signed(coeff(57)) +
signed(x_sample(56)) * signed(coeff(56)) +
signed(x_sample(55)) * signed(coeff(55)) +
signed(x_sample(54)) * signed(coeff(54)) +
```

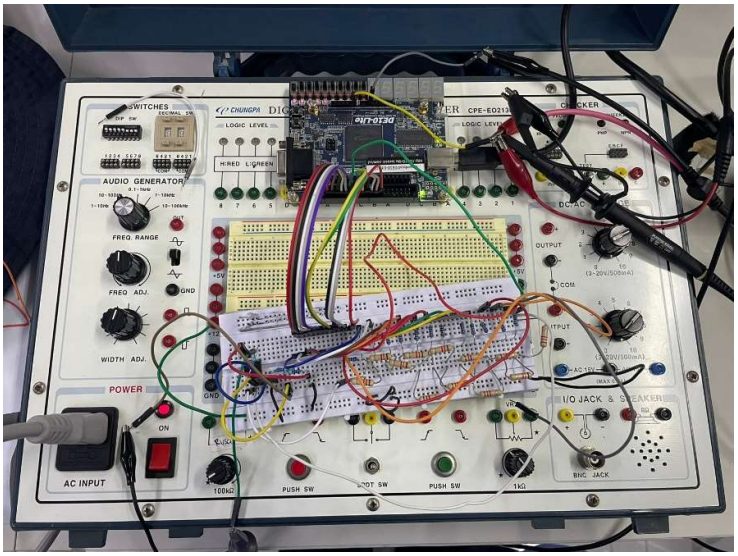
```
signed(x_sample(53)) * signed(coeff(53)) +
signed(x_sample(52)) * signed(coeff(52)) +
signed(x_sample(51)) * signed(coeff(51)) +
signed(x_sample(50)) * signed(coeff(50)) +
signed(x_sample(49)) * signed(coeff(49)) +
signed(x_sample(48)) * signed(coeff(48)) +
signed(x_sample(47)) * signed(coeff(47)) +
signed(x_sample(46)) * signed(coeff(46)) +
signed(x_sample(45)) * signed(coeff(45)) +
signed(x_sample(44)) * signed(coeff(44)) +
signed(x_sample(43)) * signed(coeff(43)) +
signed(x_sample(42)) * signed(coeff(42)) +
signed(x_sample(41)) * signed(coeff(41)) +
signed(x_sample(40)) * signed(coeff(40)) +
signed(x_sample(39)) * signed(coeff(39)) +
signed(x_sample(38)) * signed(coeff(38)) +
signed(x_sample(37)) * signed(coeff(37)) +
signed(x_sample(36)) * signed(coeff(36)) +
signed(x_sample(35)) * signed(coeff(35)) +
signed(x_sample(34)) * signed(coeff(34)) +
signed(x_sample(33)) * signed(coeff(33)) +
signed(x_sample(32)) * signed(coeff(32)) +
signed(x_sample(31)) * signed(coeff(31)) +
signed(x_sample(30)) * signed(coeff(30)) +
signed(x_sample(29)) * signed(coeff(29)) +
```

```
signed(x_sample(28)) * signed(coeff(28)) +
signed(x_sample(27)) * signed(coeff(27)) +
signed(x_sample(26)) * signed(coeff(26)) +
signed(x_sample(25)) * signed(coeff(25)) +
signed(x_sample(24)) * signed(coeff(24)) +
signed(x_sample(23)) * signed(coeff(23)) +
signed(x_sample(22)) * signed(coeff(22)) +
signed(x_sample(21)) * signed(coeff(21)) +
signed(x_sample(20)) * signed(coeff(20)) +
signed(x_sample(19)) * signed(coeff(19)) +
signed(x_sample(18)) * signed(coeff(18)) +
signed(x_sample(17)) * signed(coeff(17)) +
signed(x_sample(16)) * signed(coeff(16)) +
signed(x_sample(15)) * signed(coeff(15)) +
signed(x_sample(14)) * signed(coeff(14)) +
signed(x_sample(13)) * signed(coeff(13)) +
signed(x_sample(12)) * signed(coeff(12)) +
signed(x_sample(11)) * signed(coeff(11)) +
signed(x_sample(10)) * signed(coeff(10)) +
signed(x_sample(9)) * signed(coeff(9)) +
signed(x_sample(8)) * signed(coeff(8)) +
signed(x_sample(7)) * signed(coeff(7)) +
signed(x_sample(6)) * signed(coeff(6)) +
signed(x_sample(5)) * signed(coeff(5)) +
signed(x_sample(4)) * signed(coeff(4)) +
```

```
            signed(x_sample(3)) * signed(coeff(3)) +
            signed(x_sample(2)) * signed(coeff(2)) +
            signed(x_sample(1)) * signed(coeff(1)) +
            signed(x_sample(0)) * signed(coeff(0)));
                                    OUTPUT_FILTERED <= std_logic_vector((sum(22 downto
11))+2048);
        end if;
    end process;
end Behavioral;
```

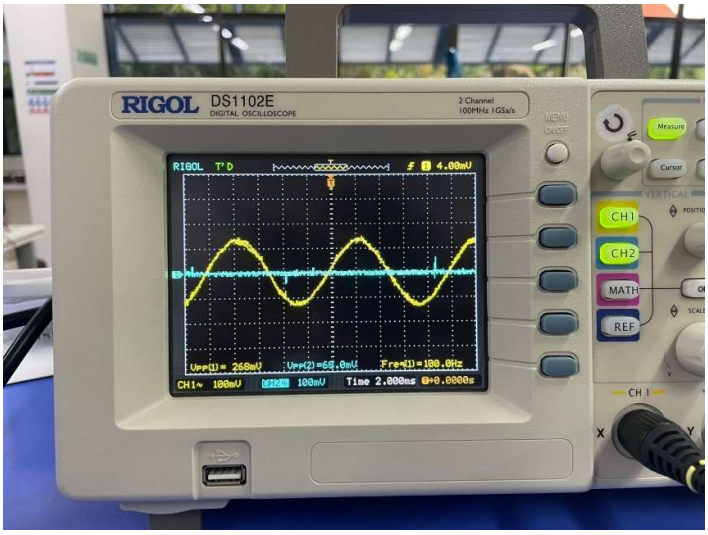
C. Implementasi Pada FPGA dan Hasil Uji Coba

Filter akan diujikan dengan sinyal input dari *function generator*, yang di sweep atau di sapu dari frekuensi 100 Hz hingga 5000 Hz. Hasil filter akan dijadikan domain kontinyu Kembali dengan DAC R2R. Berikut foto konfigurasi rangaian uji coba tersebut.

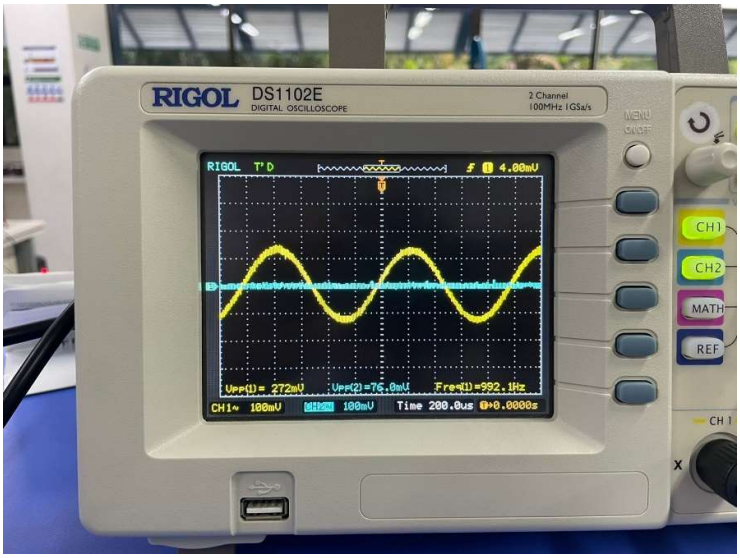


Gambar 2. Rangkaian sistem uji coba filter.

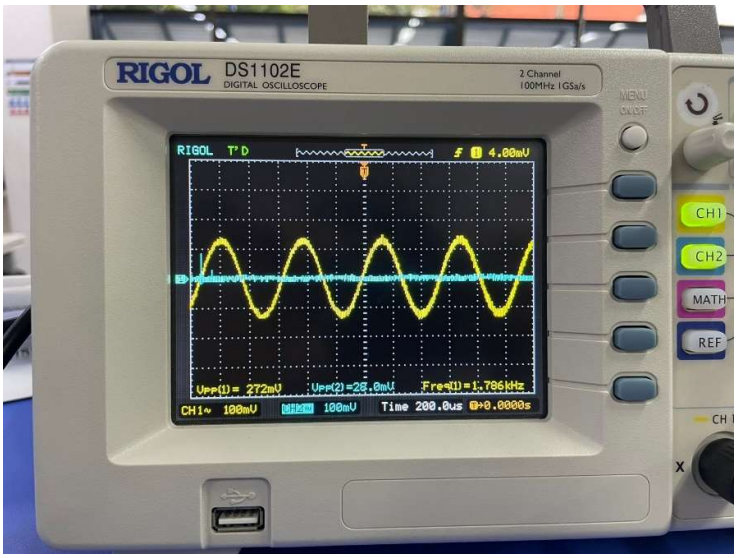
1) Hasil Uji Coba



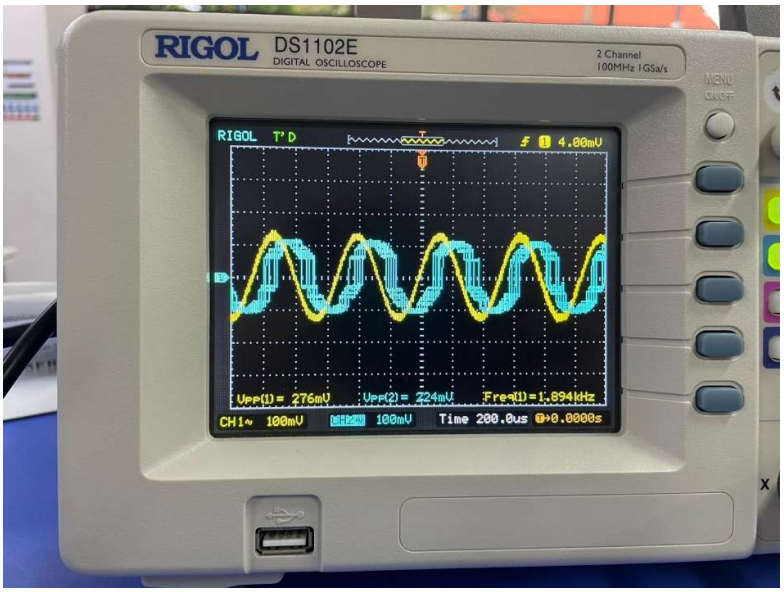
Gambar 3. Uji coba filter pada f = 100 Hz (kuning = input & biru = output).



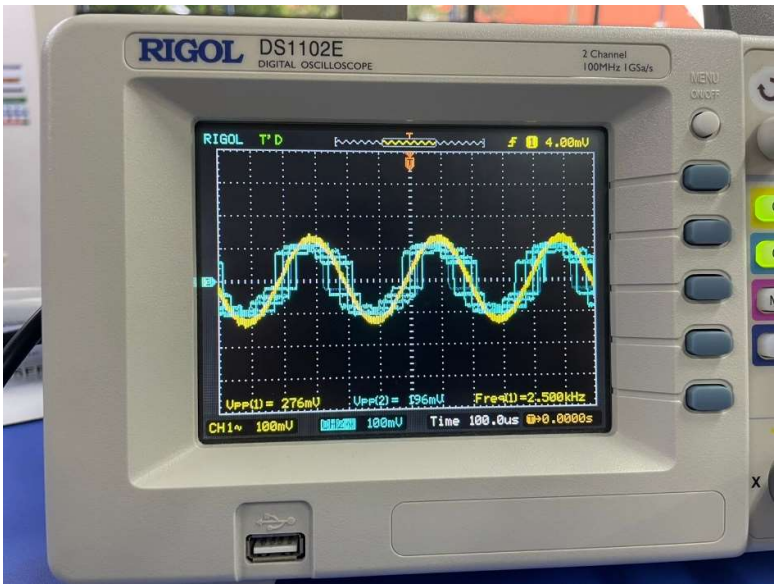
Gambar 4. Uji coba filter pada f = 1000 Hz (kuning = input & biru = output).



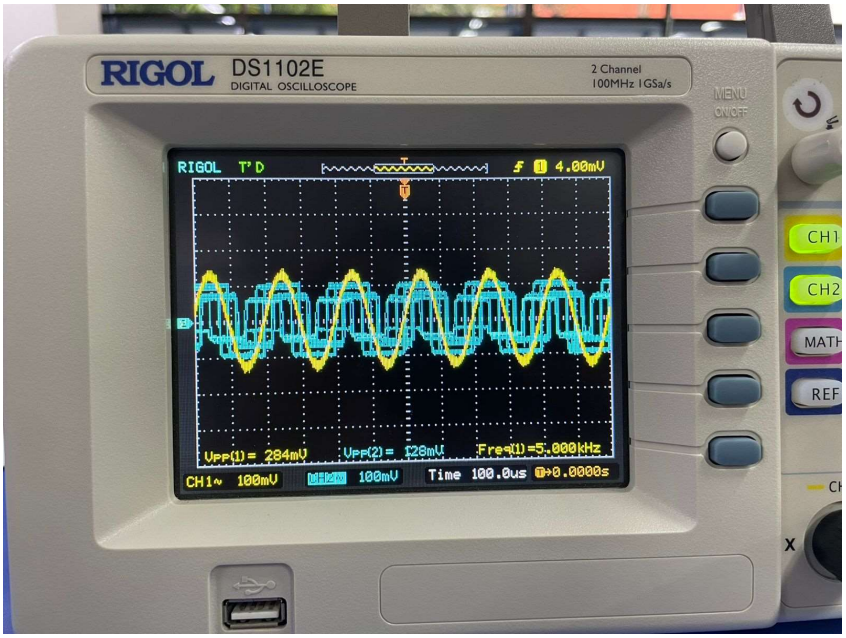
Gambar 4. Uji coba filter pada f = 1800 Hz (kuning = input & biru = output).



Gambar 4. Uji coba filter pada f = 1900 Hz (kuning = input & biru = output).



Gambar 4. Uji coba filter pada f = 2500 Hz (kuning = input & biru = output).



Gambar 4. Uji coba filter pada f = 5000 Hz (kuning = input & biru = output).

D. Kesimpulan

Melalui proses perencanaan hingga pengujian menunjukkan bahawa filter dapat berfungsi sebagaimana semestinya *high pass filter* dengan frekuensi *cut-off* 1900 Hz. Pelemahan pada frekuensi *cut-off* sebesar -3,6 dB sehingga filter berhasil mendekati -3 dB. Untuk meningkatkan kualitas hasil filter maka perlu peningkatan pada besar frekuensi sampling agar jauh lebih besar dari bandwidth sehingga output sinyal akan lebih halus tidak terlihat kasar seperti kotak-kotak.