Implementasi *Low Pass Filter* Digital IIR (*Infinite-Impulse Response*) Butterworth pada FPGA

Fikri Aulia, Mochammad Rif'an, ST., MT., dan Raden Arief Setyawan, S.T., MT.

Abstrak— **FPGA merupakan IC yang dapat diatur isi rangkaian di dalamnya. Kelebihan ini membuat FPGA mempermudah kita dalam merancang filter digital. Setiap kali kita ingin merancang filter baru, kita bisa melakukanya tanpa harus merubah atau mengganti papan rangkaian. Metode untuk merancang filter IIR yang biasa dipakai adalah metode transformasi *analog-to-digital*. Kekurangan metode tersebut adalah tidak mampu mengendalikan tanggapan fasa. AC'97 Audio Codec digunakan sebagai ADC/DAC. Sistem memiliki LUT untuk menyimpan nilai konstanta koefisien *cut-off* filter dari frekuensi 10.000Hz hingga 20.000Hz.**

Prosentase slice dan LUT yang dipakai untuk implementasi hanya 3%. FPGA lebih banyak menggunakan komponen pengali dan penjumlah. Hasil pengujian spektrum menunjukan bahwa sinyal keluaran mengalami penurunan jauh sebelum frekuensi *cut-off*. Pengujian filter pada domain waktu menunjukan sistem mampu memfilter sinyal dengan frekuensi *cut-off* berbeda-beda, namun sinyal keluaran memiliki fasa tidak sama.

Kata kunci—**FPGA, Filter IIR, AC'97 Audio Codec**

I. PENDAHULUAN

Teknologi digital saat ini menjadi teknologi yang berkembang pesat dan mulai banyak menggantikan teknologi analog. Salah satu bentuk pengolahan sinyal digital adalah filter. Pengimplementasian filter digital bisa dilakukan melalui algoritma dan dijalankan di komputer ataupun melalui rangkaian. Filter yang terbuat dari rangkaian tidak perlu terbebani oleh perhitungan, namun tidak bisa atau sulit diubah spesifikasinya.

Dibandingkan dengan filter FIR, implementasi filter IIR membutuhkan lebih sedikit parameter, sehingga lebih sedikit memori yang digunakan.

FPGA merupakan IC yang dapat diatur isi rangkaian di dalamnya. Kelebihan ini membuat FPGA mempermudah kita dalam merancang filter digital. Setiap kali kita ingin merancang filter baru, kita bisa melakukanya tanpa harus merubah atau mengganti papan rangkaian.

Berdasarkan keunggulan-keunggulan tersebut, maka penulis tertarik untuk emelakukan penelitian tentang pengimplementasian filter digital IIR ke dalam perangkat keras FPGA ini.

Fikri Aulia adalah mahasiswa Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya Malang Indonesia (Korespondensi penulis melalui HP 08123317610; email micro18fsystem@gmail.com)

Mochammad Rif'an adalah dosen Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya Malang Indonesia.

Raden Arief Setyawan adalah dosen Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya Malang Indonesia

II. PERANCANGAN DAN PEMBUATAN ALAT

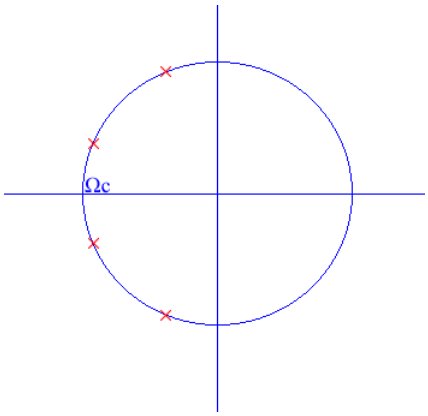
A. Perancangan Filter

Perancangan filter berguna untuk menentukan konstanta pengali filter untuk masing-masing frekuensi cut-off. Metode perancangan filter IIR digital yang digunakan adalah metode transformasi A/D(analog-to-digital)[1]. Frekuensi sampling ADC/DAC yang digunakan adalah 48.000Hz. Metode ini banyak diketahui namun tidak dapat mengendalikan karakteristik fasa filter. Frekuensi cut-off filter dapat diubah dari 10.000Hz hingga 20.000Hz dengan selisih 1.000Hz.

Fungsi alih filter IIR untuk perancangan ini adalah.

$$H(s)=\frac{\Omega^4}{\prod_{n=1}^{4}(s-pn)} \quad \text{.....(1)}$$

Gambar 1 menunjukan letak tiang untuk filter orde 4.



Gambar 1. Letak Tiang Filter Orde 4

Mengacu pada Gambar 1 diatas, didapatkan nilai tiang sebagai berikut:

$$p_1=\Omega_c(cos(112{,}5^o)+j\,sin(112{,}5^o)) \quad \text{....(2)}$$

$$p_2=\Omega_c(cos(157{,}5^o)+j\,sin(157{,}5^o)) \quad \text{....(3)}$$

$$p_3=\Omega_c(cos(202{,}5^o)+j\,sin(202{,}5^o)) \quad \text{....(4)}$$

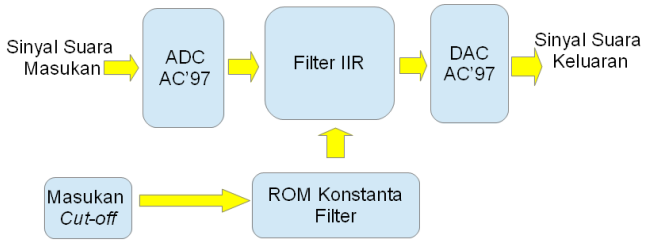
$$p_4=\Omega_c(cos(247{,}5^o)+j\,sin(247{,}5^o)) \quad \text{....(5)}$$

Fungsi alih filter analog kemudian ditransformasikan menjadi digital menggunakan transformasi bilinear[2]. Hasil transformasi bilinear adalah konstanta filter berbentuk pecahan desimal.

Konstanta filter kemudian diubah menjadi bentuk signed fixed-point agar dapat diolah dalam FPGA. Cara merubah adalah mengalikan dengan 2^{28}, lalu dikonversi menjadi bentuk biner. Konstanta negatif direpresentasikan menggunakan format *two-complement*[3].

B. Blok Diagram Sistem

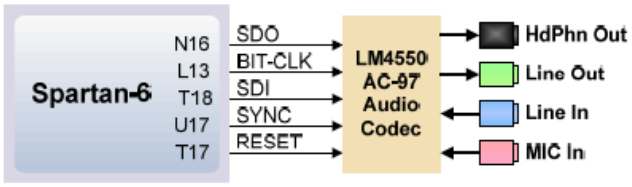
Implementasi dimulai dengan membuat blok diagram sistem. Gambar 2 menunjukan blok diagram sistem.



Gambar 2. Blok Diagram Sistem

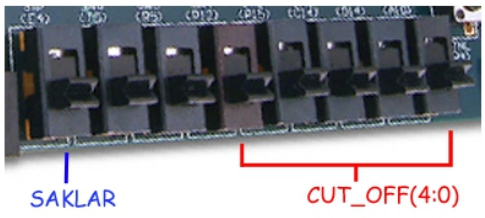
C. Input/Output Sistem

Input sinyal FPGA adalah Line In AC-97 codec. Sedangkan *output* sinyal adalah Line Out. Gambar 3 Menunjukan letak *input* dan *output* sinyal[4].

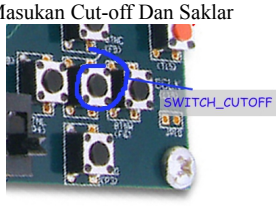


Gambar 3. Input Output Sinyal
Sumber: Atlyst Reference Manual

Saklar dan tombol pada modul berguna sebagai pengendali frekuensi *cut-off*. Gambar 4 dan Gambar 5 menunjukan komponen *input*.



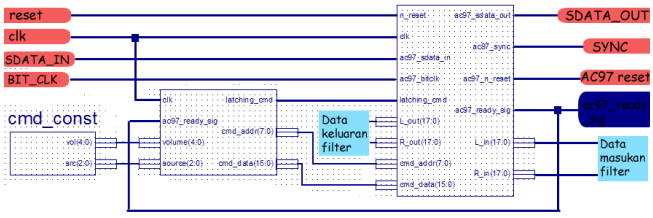
Gambar 4. Masukan Cut-off Dan Saklar



Gambar 5. Tombol Reset Dan Pemicu Cut-off

D. AC'97 Hardware Driver

AC'97 memiliki keluaran dan masukan serial[5], sehingga diperlukan blok komponen untuk mengubah keluaran dan masukan tersebut agar dapat diproses oleh filter. Rancangan hardware driver berdasarkan desain dari eewiki[6]. Gambar 6 menunjukan konfigurasi sambungan AC'97 Hardware Driver



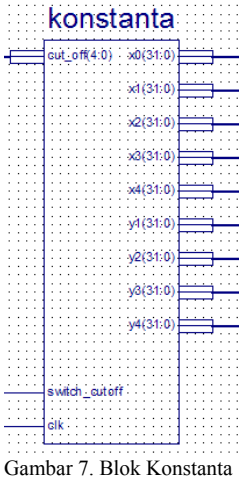
Gambar 6. Konfigurasi Sambungan AC'97 Hardware Driver

Hardware driver terdiri dari tiga komponen yaitu AC97 dan AC97cmd. Penjelasan mengenai fungsi masing-masing komponen adalah sebagai berikut:

1) Blok AC97cmd berfungsi sebagai pengatur register dan data yang akan dikirim ke ADC/DAC.
2) Blok AC97 berfungsi sebagai enkoder dan dekoder ADC/DAC.
3) cmd_const agar AC'97 menggunakan masukan hanya dari line-in dan atenuasi yang tetap. Masukan source harus bernilai B"100" dan volume bernilai B"00000".

E. Blok Konstanta Filter

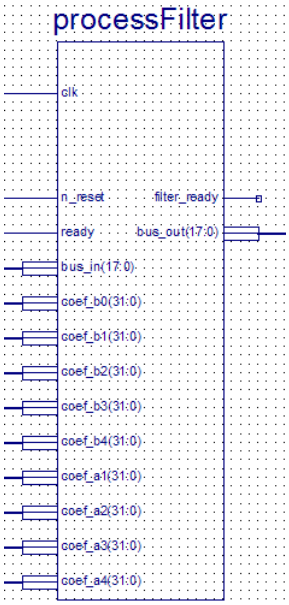
Blok konstanta filter berisi LUT(Look-Up Table) dan multiplexer. LUT berguna untuk menyimpan nilai masing-masing konstanta filter. Input blok konstanta filter akan mengatur tabel konstanta yang menjadi keluaran. Saklar cut_off(4:0) berfungsi untuk mengatur frekuensi *cut-off*. Tombol SWITCH_CUTOFF akan memicu perubahan keluaran blok ini.



Gambar 7. Blok Konstanta

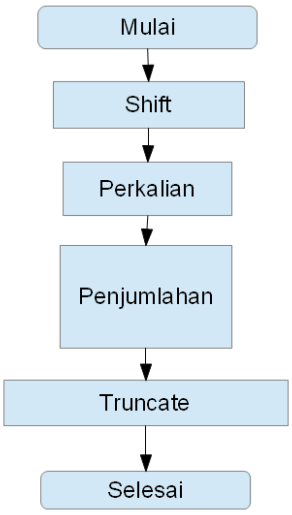
F. Proses Filter

Blok Proses Filter berfungsi untuk memproses sinyal masukan. Blok ini mengambil tabel konstanta dari Blok Konstanta Filter. Gambar 8 menunjukan blok Proses Filter.



Gambar 8. Blok Proses Filter

Gambar 9 menunjukan flowchart algoritma pemprosesan filter pada FPGA.



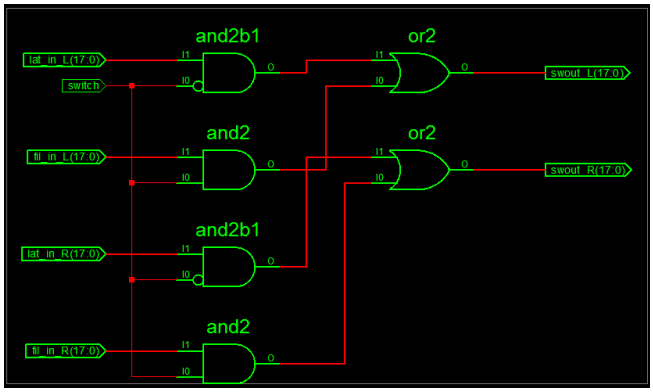
Gambar 9. Flowchart Proses Filter

G. Latch

Blok Latch berfungsi melewatkan data tanpa melakukan proses. Blok sistem ini berguna agar sinyal asli dapat diperhatikan tanpa perlu memprogram ulang FPGA.

H. Switch

Blok Switch berguna untuk memilih sinyal dari Proses Filter atau dari Latch yang akan dikirimkan ke DAC. Blok ini berupa multiplexer. Gambar 10 Menunjukan skematik blok Switch



Gambar 10. Skematik Blok Switch

III. PENGUJIAN DAN ANALISIS

A. Pengujian Hardware FPGA

Pengujian hardware FPGA bertujuan untuk mengetahui berapa banyak slice dan komponen FPGA yang telah terpakai. Hasil pengujian ditunjukan pada Tabel 1.

TABEL 1.
DEVICE UTILIZATION SUMMARY

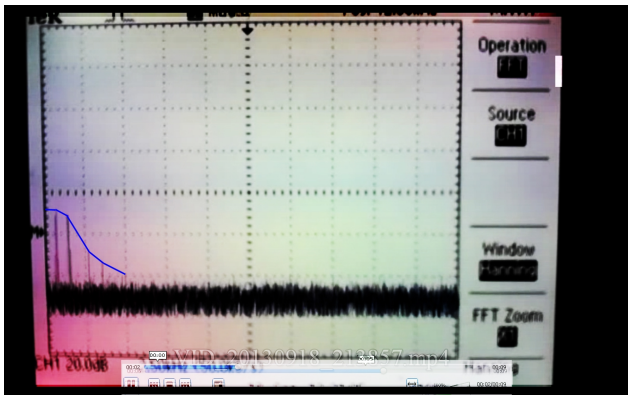
No.	Slice Logic Utilization	Used	Available	Utilization
1	Number Of Slice Registers	1708	54576	3.00%
2	Number Used As Flip Flops	622		
3	Number Used As Latches	828		
4	Number Used As Latch-thrus	0		
5	Number Used As AND/OR Logics	258		
6	Number Of Slice LUTs	882	27288	3.00%
7	Number Used As Logic	693	27288	2.00%
8	Number Of Occupied Slices	464	6822	6.00%
9	Nummber Of MUXCYs Used	504	13644	3.00%

10	Number Of LUT Flip Flop Pairs Used	1657		
11	Number With An Unused Flip Flop	157	1657	9.00%
12	Number With An Unused LUT	775	1657	46.00%
13	Number Of Fully Used LUT-FF Pairs	725	1657	43.00%
14	" Number Of Slice Register Sites Lost to Control Set Restrictions"	46	54576	1.00%
15	Number Of Bonded IOBs	14	218	6.00%
16	Number Of DSP48A1s	36	58	62.00%

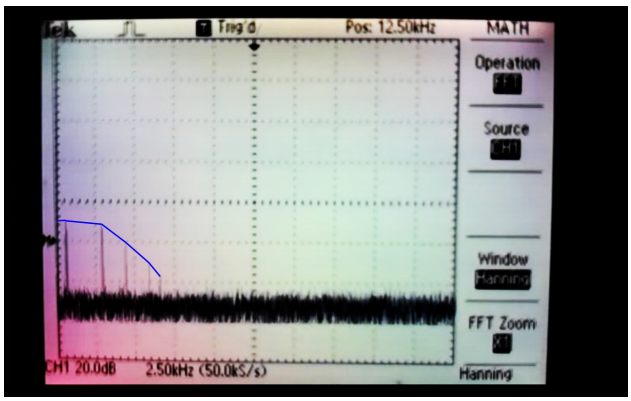
Dari data diatas dapat diketahui bahwa implementasi filter pada FPGA menggunakan jumlah slice yang sangat sedikit. Namun komponen aritmatik DSP48A1[7] yang terpakai dalam sistem ini mencapai 36 unit. Jumlah tersebut mencapai lebih dari setengah dari total unit DSP48A1.

B. Pengujian Magnitude Frekuensi Sinyal Keluaran

Pengujian sinyal keluaran bertujuan untuk mengetahui bentuk sinyal keluaran di domain frekuensi. Sinyal masukan dalam pengujian ini menggunakan frekuensi yang bertambah dari 20Hz hingga 22.000Hz. Gambar 11 dan Gambar 12 menunjukan hasil pengujian magnitude frekuensi sinyal keluaran.



Gambar 11. Pengujian Frekuensi Cut-off 10.000Hz

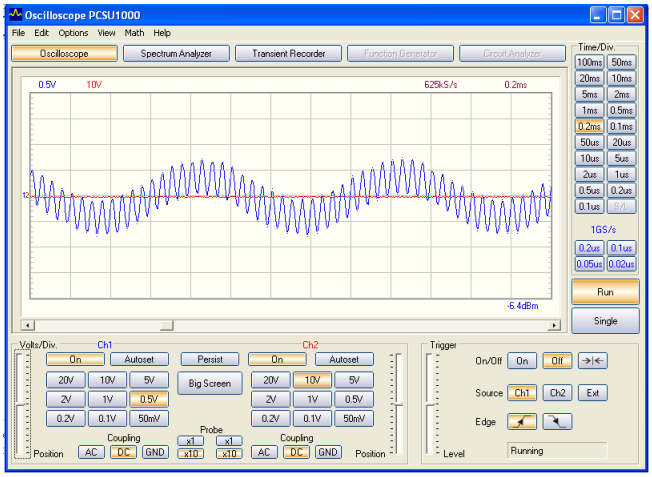


Gambar 12.Pengujian Frekuensi Cut-off 20.000Hz

Hasil pengujian menunjukan sinyal keluaran mengalami penurunan jauh sebelum frekuensi cut-off. Dari pengujian tersebut dapat disimpulkan bahwa frekuensi sinyal keluaran filter orde 4 sangat landai.

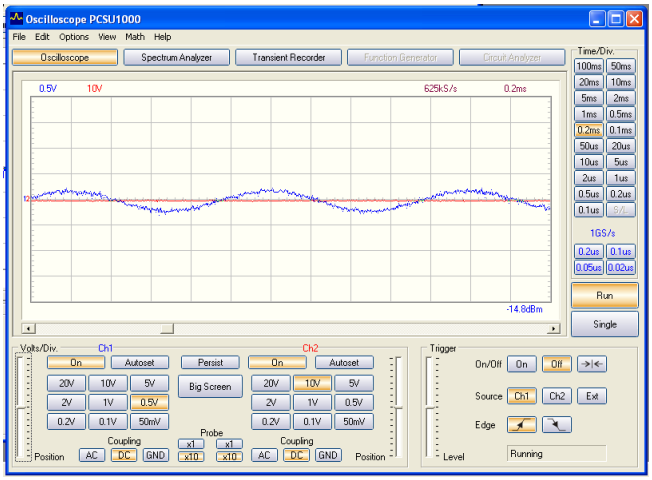
C. Pengujian Magnitude Frekuensi Campuran

Pengujian sinyal keluaran bertujuan untuk mengetahui bentuk sinyal keluaran di domain waktu. Pengujian ini menggunakan frekuensi 1.000Hz dan 20.000Hz yang dicampur. Gambar 13 menunjukan bentuk sinyal masukan.

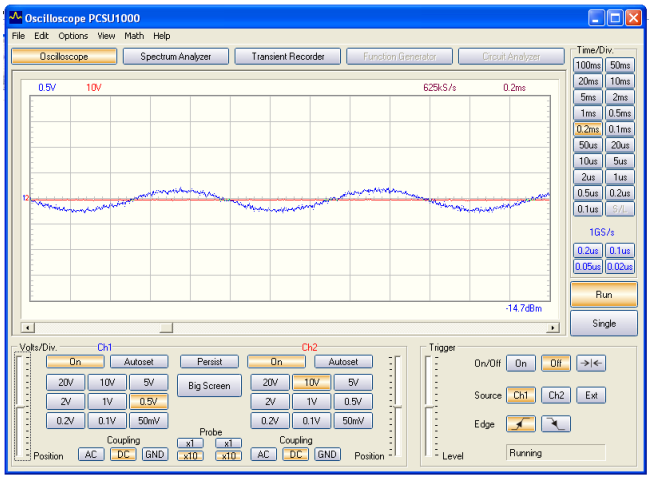


Gambar 13. Gambar Sinyal Frekuensi 1.000Hz Dan 20.000Hz

Gambar 14 dan Gambar 15 menunjukan hasil pengujian untuk frekuensi cut-off 12.000Hz dan 16.000Hz.



Gambar 14. Gambar Sinyal Frekuensi Cut-off 12.000Hz



Gambar 15. Gambar Sinyal Frekuensi Cut-off 16.000Hz

Hasil pengujian menunjukan filter yang diimplementasikan ke dalam FPGA mampu memfilter sinyal campuran. Sinyal keluaran untuk masing-masing frekuensi *cut-off* memiliki fasa yang berbeda-beda.

IV. KESIMPULAN DAN SARAN

A. Kesimpulan

Berdasarkan pengujian, analisis dan tinjauan pustaka ini dapat ditarik kesimpulan sebagai berikut.

1. Low pass filter IIR Butterworth orde 4 dapat dirancang menggunakan metode *A/D Filter Transformation*. Metode tersebut men-transformasikan filter analog menjadi digital dengan cara mengubah domain s menjadi domain z. Salah satu cara transformasi domain adalah Transformasi Bilinear. Metode ini memiliki spektrum sinyal keluaran yang mengalami penurunan sebelum mencapai frekuensi *cutoff*.
2. Implementasi Filter IIR orde 4 pada FPGA membutuhkan slice register dan LUT sebanyak 3% dari keseluruhan. FPGA mampu mem-filter frekuensi diatas frekuensi *cutoff* dengan baik.

B. Saran

Saran-saran dalam pengimplementasian maupun peningkatan unjuk kerja sistem ini dapat diuraikan sebagai berikut:

1. Menggunakan frekuensi kerja filter yang lebih tinggi.
2. Implementasi filter IIR orde 4 hanya memakai sedikit slice FPGA. Oleh karena itu pada pengembangan berikutnya disarankan untuk mendesain filter IIR dengan orde yang lebih tinggi.

DAFTAR PUSTAKA

[1] Ingle, Vinay K & Proakis, John G. 1999. *Digital Signal Processing Using MATLAB V 4*. Boston: PWS Publishing Company.

[2] Karris, Steven T. 2007. *Signals and Systems with MATLAB Computing and Simulink Modeling, Third Edition*. United States of America: Orchad Publication.

[3] Bease-Uwe Meyer. 2007. *Digital Signal Processing with Field Programmable Gate Arrays*. Berlin: Spinger.

[4] Digilent. *Atlys Board Reference Manual*. www.digilentinc.com.

[5] National Semiconductor. *Datasheet of LM4550*. www.national.com.

[6] Storey, Tony. 2013. *AC'97 Codec Hardware Driver Example*. http://eewiki.net/display/LOGIC/AC%2797+Codec+Hardware+Driver+Example.

[7] Xilinx. 2009. *User Guide of Spartan-6 FPGA DSP48A1 Slice*. www.xilinx.com.

Fikri Aulia, Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya, Juli 2013, Implementasi *Low Pass Filter* Digital IIR (*Infinite-Impulse Response*) Butterworth pada FPGA Dosen Pembimbing: Mochammad Rif'an, ST., MT. Dan Raden Arief Setyawan ST., MT.