PERANCANGAN HISTOGRAM DENGAN MENERAPKAN PERSAMAAN MAGNITUDE HASIL KELUARAN FFT 256 POINT PADA TAMPILAN VGA MONITOR PERANGKAT AUDIO SPECTRUM ANALYZER BERBASIS FPGA

Bernardus Galih Dwi Wicaksono[1], Gde KM Atmajaya[1], Rudi Uswarman[1], Harry Yuliansyah[1], Arif Sasongko [1,2]

[1]Program Studi Teknik Elektro, Institut Teknologi Sumatera (ITERA), Lampung Selatan, Indonesia

[1]Program Studi Teknik Elektro, Institut Teknologi Bandung (ITB), Bandung, Indonesia

b.galihdwi@gmail.com, gde_komang94@yahoo.com, uswarman@itera.ac.id, harry@itera.ac.id, *asasongko@stei.itb.ac.id*

***Abstrak*— Kebutuhan akan perangkat yang mampu melakukan transformasi sinyal audio dari domain waktu menjadi domain frekuensi menjadi tahap awal untuk mengetahui informasi dari sinyal audio. Perangkat *Audio Spectrum Analyzer* merupakan jawaban atas kebutuhan tersebut. Perangkat ini diimplementasikan ke *Field Programmable Gate Array* (FPGA) untuk memproses data masukkan berupa sinyal audio. Data hasil transformasi sinyal audio akan ditampilkan pada *Video Graphics Array* (VGA) monitor dalam bentuk histogram. Tugas akhir ini menjabarkan perancangan, implementasi dan pengujian dalam tampilan perangkat Audio Spectrum Analyzer berbasis FPGA pada VGA monitor. Perancangan menggunakan 3 blok utama yakni blok persamaan *magnitude*, blok memori *magnitude*, dan blok grafis VGA. Blok persamaan *magnitude* berfungsi untuk melakukan operasi bilangan berdasarkan persamaan *magnitude*. Nilai masukkan yang digunakan merupakan hasil keluaran *Fast Fourier Transform* (FFT) berupa bilangan real dan imajiner bertipe *unsigned integer*. Blok memori *magnitude* berfungsi untuk menyimpan keluaran dari blok persamaan *magnitude* berupa bilangan bertipe *unsigned integer* sebagai nilai spektrum. Dalam blok memori *magnitude*, terdapat kontroler untuk melakukan kontrol terhadap nilai yang di simpan di dalam memori *magnitude*. Nilai yang telah di simpan lalu di interaksikan dengan blok VGA yang berfungsi untuk menampilkan data yang telah diproses dalam bentuk histogram. Implementasi dilakukan secara bersamaan untuk menampilkan data yang disimpan ke VGA monitor. Hasil pengujian berupa histogram yang merepresentasikan nilai spektrum hasil kalkulasi menggunakan persamaan *magnitude*. Tampilan histogram pada layar VGA monitor bersesuaian dengan sinyal masukkan yang diterima.**

Kata kunci — ***FFT, Persamaan Magnitude, Spektrum, Histogram, VGA Monitor, FPGA***

I. PENDAHULUAN

Sinyal audio (gelombang suara) merupakan suatu gelombang yang dihasilkan pada rentang frekuensi yang mampu didengar oleh manusia yakni antara 20 Hz hingga 20 KHz [1]. Penerapan terhadap sinyal audio telah mampu di aplikasikan di berbagai bidang pada kehidupan manusia seperti pengolahan suara dan nada pada bidang suara dan musik, pengolahan citra dan audio pada bidang video dan gambar, pendeteksian gejala penyakit pada bidang biomedis, dan berbagai penerapan di bidang lainnya. Di tinjau dari pengaplikasian tersebut maka penelitian dengan menerapkan sinyal audio telah banyak dikembangkan. Penelitian terhadap sinyal audio telah dimulai dari tahap pembelajaran di sekolah menengah atas, perguruan tinggi hingga riset ahli. Dalam melakukan penelitian, para peneliti mengumpulkan informasi – informasi terkait sinyal audio agar dapat di analisa dan dikembangkan. Salah satu informasi yang dibutuhkan terkait sinyal audio yakni frekuensi dari sinyal audio tersebut.

Perangkat yang mampu memberikan informasi terkait frekuensi sinyal audio adalah *Audio Spectrum Analyzer*. Perangkat ini mampu melakukan transformasi sinyal dari domain waktu menjadi domain frekuensi. Pengimplementasian terhadap perangkat ini dapat dalam bentuk perangkat lunak (*software*) maupun perangkat keras (*hardware*). Dalam implementasi menggunakan perangkat lunak, penggunaan algoritma dan hasil pengimplementasinya dapat dilakukan dengan mudah tanpa perlu memerhatikan perilaku dari perangkat lunak tersebut. Berbeda dengan implementasi menggunakan perangkat keras, penggunaan algoritma harus menyesuaikan dengan perilaku perangkat keras. Implementasi dalam perangkat keras mampu memberikan performa kecepatan yang lebih cepat dari pada implementasi dalam perangkat lunak [2]. Hal tersebut terjadi salah satunya dikarenakan dalam melakukan implementasi algoritma disesuaikan dengan perilaku perangkat yang digunakan, sehingga interaksi antara algoritma dan perangkat bersesuaian.

Performa kecepatan pada implementasi perangkat keras menjadi keunggulan dalam merancang *Audio Spectrum Analyzer*. Perangkat keras yang mampu diimplementasikan salah satunya yakni *Field Programmable Gate Array* (FPGA). Perangkat ini mampu melakukan pemrosesan sinyal hingga 100 MHz [3]. Pengembangan terhadap perangkat *Audio Spectrum Analyzer* berbasis perangkat keras FPGA telah banyak dilakukan, namun pengimplementasiannya menggunakan board FPGA yang cukup mahal dan algoritma yang cukup rumit[10-11]. Dalam tugas akhir ini, implementasi dilakukan pada board FPGA Altera DE-1 yakni salah satu *board* FPGA yang relatif lebih murah dibandingkan dengan *board* FPGA lain dan dengan menerapkan algoritma *Fast Fourier Transform* (FFT) yakni algoritma yang melakukan transformasi domain waktu ke domain frekuensi. Algoritma FFT yang digunakan yakni 256 poin. Untuk penyajian dari hasil algoritma tersebut, digunakan tampilan histogram pada *Video Graphics Array* (VGA) monitor. Histogram dirancang dengan menerapkan algoritma persamaan *magnitude* yang kemudian akan memunculkan nilai spektrum sebagai nilai yang merepresentasikan histogram. Penampilan menggunakan layar VGA monitor digunakan untuk mempermudah dalam pembacaan dan analisa terhadap frekuensi sinyal audio. Selain

itu penggunaan VGA monitor dipilih karena VGA monitor sangat mudah ditemui dalam khususnya bagi peneliti di dalam laboratorium ataupun tempat penelitian. Dengan adanya perangkat *Audio Spectrum Analyzer* berbasis FPGA hasil dari tugas akhir ini diharapkan mampu membantu dan memudahkan dalam memperoleh informasi (khususnya frekuensi) terkait penelitian terhadap sinyal audio.

II. SPESIFIKASI

Spesifikasi rancangan sistem penampilan pada perangkat *Audio Spectrum Analyzer* sebagai berikut :

- Sinyal masukan menggunakan sinyal audio
  Sinyal masukkan yang digunakan merupakan sinyal audio (gelombang suara) memiliki rentang frekuensi 20 Hz – 20 KHz.
- Lebar setiap data sampel adalah 16 bit
  Lebar setiap data sampel dipilih 16 bit karena kepresisian data yang cukup baik dan juga nilai data yang tidak terlalu besar. Nilai ini dipilih berdasarkan konfigurasi *Audio Codec* yakni 8, 16, 24 dan 32.
- Frekuensi sampling yang digunakan adalah 48 KHz
  Frekuensi sampling ditentukan menggunakan metode Nyquist [7] dengan perhitungan sebagai berikut:
  $$fs = Max\ Freq * 2$$
  $$fs = 40.000\ Hz$$
  Hasil frekuensi tersebut tidak dapat digunakan karena pada *chip Audio Codec* hanya terdapat konfigurasi frekuensi sampling sebesar 48 KHz, sehingga nilai frekuensi sampling yang digunakan 48 KHz.
- Frekuensi bin yang digunakan 188
  Frekuensi bin didapat dari perhitungan frekuensi sampling dan banyak sampling yaitu 256 [7]. Berikut ini perhitungan frekuensi bin:
  $$Frequency\ Resolution = \frac{Fs}{N}$$
  $$Frequency\ Resolution = \frac{48.000}{256}$$
  $$Frequency\ Resolution = 187{,}5\ \approx 188\frac{Fs}{N}$$
  Nilai dari frekuensi bin menjadi acuan nilai untuk 1 data histogram
- Jumlah poin FFT digunakan adalah 256 poin
  Penentuan terhadap jumlah FFT poin berdasarkan dari banyaknya sampel yang dicuplik dan disimpan pada memori.
- Jumlah penampilan data spektrum adalah 128
  Jumlah penampilan data spektrum ditentukan dari jumlah data sampling yang digunakan yang dibagi dua, hal ini dikarenakan terdapat mirroring pada setengah data tersebut.
- Besar Memori *Magnitude* adalah 256 Byte
  Memori digunakan untuk menyimpan data dari hasil persamaan *magnitude* untuk proses selanjutnya ke VGA [8]. Perhitungan terhadap besar nilai memori yang digunakan adalah sebagai berikut :
  $$Total\ Memori = 16\ bit\ x\ 128$$
  $$Total\ Memori = 2.048\ bit$$
  $$Total\ Memori = 256\ Byte$$
- Resolusi tampilan 640x480
  Resolusi ditentukan berdasarkan format standar dalam penampilan pada layar monitor VGA yakni 640 x 480 dengan refresh rate 60 Hz [9].

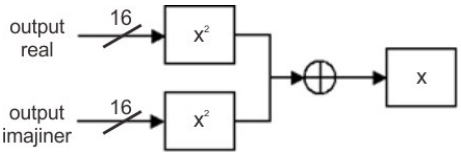
III. PERANCANGAN DAN IMPLEMENTASI

1. Garis Besar Perancangan Sistem

Gambar 1. Garis besar perancangan sistem.

Pada gambar diatas FFT *processor* merupakan proses perhitungan FFT pada 256 data yang kemudian memiliki keluaran berupa bilangan kompleks real dan imajiner. Hasil keluaran FFT *processor* kemudian diproses menggunakan operasi persamaan *magnitude* untuk memperoleh nilai *magnitude* dalam bentuk polar. Nilai tersebut kemudian disimpan ke dalam memori. Nilai yang disimpan dalam memori kemudian disinkronkan dengan VGA *controller* untuk dapat memberikan tampilan pada layar monitor berupa tampilan histogram.

2. Rangkaian Persamaan *Magnitude*



Gambar 2. Diagram blok persamaan *magnitude*.

Penerapan diagram blok *magnitude* pada keluaran blok FFT akan menggunakan 3 blok komponen *megafunction* yang tersedia pada board FPGA Altera.

Tabel 1. Detail blok komponen *multiplier*.

Blok	*LPM_MULT(Multiplier) IP Core*
Fungsi	Melakukan operasi kuadrat
Input	*Output* real dan *output* imajiner *Unsigned integer* 16 bit
Output	Data real kuadrat dan data imajiner kuadrat *Unsigned integer* 32 bit

Tabel 2. Detail blok komponen *adder*.

Blok	*LPM_ADD_SUB*
Fungsi	Melakukan operasi penjumlahan
Input	Data real kuadrat dan data imajiner kuadrat

	Unsigned integer 32 bit
Output	Data hasil penjumlahan real dan imajiner *Unsigned integer* 32 bit

Tabel 3. Detail blok komponen square root.

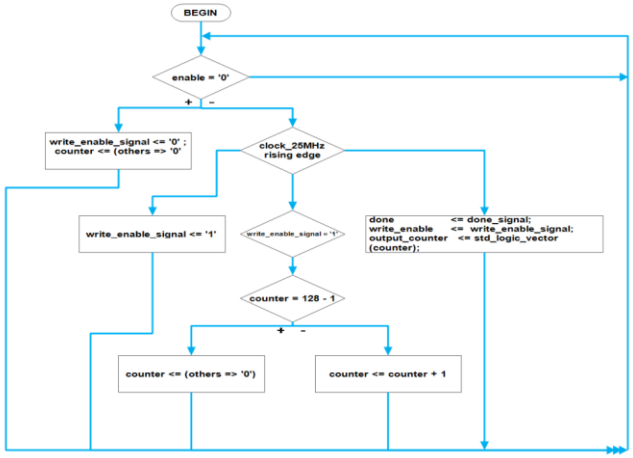
Blok	*ALTSQRT* (*Integer Square Root*) *IP Core*
Fungsi	Melakukan operasi akar kuadrat
Input	Data hasil penjumlahan real dan imajiner *Unsigned integer* 32 bit
Output	Nilai *magnitude* data hasil operasi akar kuadrat *Unsigned integer* 16 bit

3. Memori *Magnitude*

Blok memori digunakan untuk menyimpan data yang dihasilkan oleh persamaan *magnitude*. Blok memori akan menggunakan tipe dual port *Random Access Memory* (RAM) yang telah disediakan dalam *megafunction board* FPGA. . Besar memori yang digunakan berhubungan dengan jebar data dan jumlah data yang akan ditampung, sehingga :

$$\text{Total Memori} = 16 \text{ bit x } 128 \text{ poin} = 2.048 \text{ bit} = 256 \text{ Byte}$$

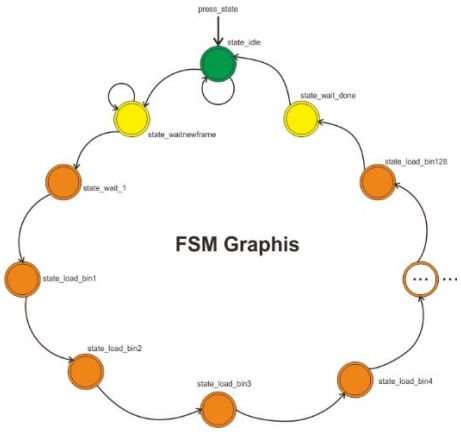
Dalam penggunaan RAM diperlukan kontroler yang akan mengidentifikasi data untuk memenuhi syarat. Penggunaan kontroler RAM ini sebagai kendali untuk menulis dan membaca data pada RAM.



Gambar 3. *Flowchart* kontroler RAM.

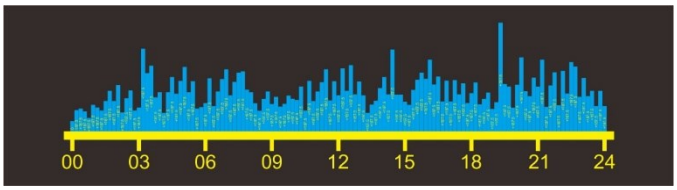
4. Tampilan VGA

Perancangan tampilan VGA diperlukan untuk melakukan mapping terhadap tampilan yang akan berinteraksi secara langsung dengan user. Resolusi yang digunakan untuk penampilan pada VGA monitor adalah 640 x 480 @60Hz, hal ini ditentukan berdasarkan format standar dalam penampilan VGA monitor. Dalam pengimplementasian VGA, data yang akan digunakan sebagai spektrum menerapkan *Finite State Machine* (FSM) berikut :



Gambar 4. FSM graphis.

Berdasarkan FSM tersebut, perancangan dan pengimplementasian tampilan pada VGA Monitor disusun sebagai berikut :



Gambar 5. Desain dan implementasi tampilan VGA monitor.

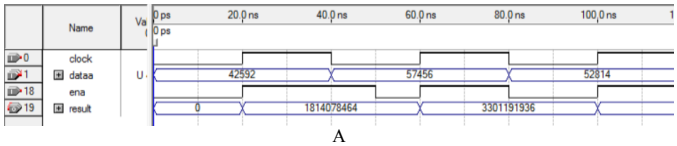
Setiap nilai histogram mewaikili nilai 187,5 Hz dan kelipatannya. Hal ini diperoleh dari perhitungan terhadap jumlah frekuensi sampling dan banyaknya data yang akan ditampilkan.

IV. PENGUJIAN DAN VERIFIKASI

1. Pengujian Rangkaian Persamaan *Magnitude*

Pengujian terhadap rangkaian persamaan *magnitude* dilakukan secara terpisah dalam satu kesatuan persamaan. Pengujian ini dibagi menjadi 3 tahap yakni tahap awal (*multiplier*), tahap tengah (*adder*), dan tahap akhir (*square root*).

- **Tahap Awal (*multiplier*)**
  Pengujian dilakukan menggunakan data masukkan yang telah ditentukan, lalu data keluaran dari blok komponen multiplier dibandingkan dengan data perhitungan secara manual.



A

B

Gambar 6. (A) dan (B) Pengujian Kompoenen *Multiplier*.

Dari hasil pengujian terlihat bahwa data masukkan yakni dataa merupakan data yang akan dilakukan operasi kuadrat. Data masukkan diproses ketika clock menunjukkan `rising_edge` dan nilai `enable = 1`. Ketika data masukkan tidak dalam kondisi `rising_edge` dan nilai `enable = 1`, maka data yang digunakan adalah data sebelumnya. Berikut merupakan data hasil perhitungan manual operasi kuadrat.

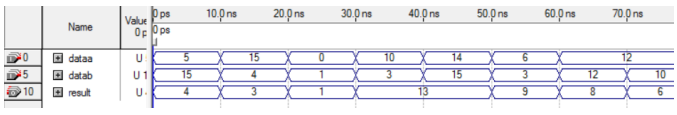
Tabel 4. Hasil perhitungan manual operasi kuadrat.

Operasi Kuadrat	
Data Masukkan	Data Keluaran
42592	1814078464
57456	3301191936
52814	2789318596
40755	1660970025
47452	2251692304
5021	25210441
56856	3232604736

Terlihat bahwa 7 data masukkan yang seharusnya bernilai sama dengan tabel diatas hanya menghasilkan 4 data keluaran yang mana 3 data masukkan lainnya tidak diproses karena tidak memenuhi kondisi. Dari pengujian ini dapat dikatakan blok komponen telah berjalan sesuai dengan operasi kuadrat dan melakukan operasi sesuai dengan kondisi tertentu, hal ini bersesuaian dengan implementasi pada perangkat *Audio Spectrum Analyzer* Berbasis FPGA.

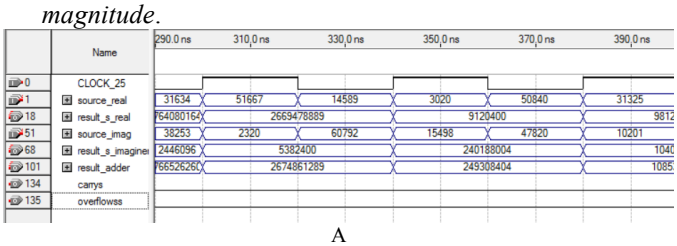
- **Tahap Tengah (*adder*)**

Pengujian terhadap blok komponen adder dengan menggunakan 2 data masukkan dari perhitungan kuadrat yakni data real dan imajiner. Data keluaran blok komponen adder kemudian dibandingkan dengan data perhitungan secara manual.

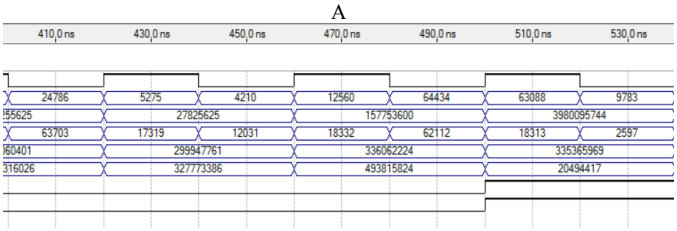


Gambar 7. Pengujian Kompoenen *Adder* 16 bit.

Dari hasil simulasi terlihat bahwa blok komponen adder bekerja tanpa harus memenuhi suatu kondisi, dengan kata lain blok bekerja terhadap setiap data masukkan yang diterima. Dari simulasi tersebut kemudian diimplementasikan pada blok satu kesatuan dengan blok *magnitude*.



A



B

Gambar 8. (A) dan (B) Pengujian Kompoenen *Adder* 32 bit.

Tabel 5. Hasil perhitungan manual operasi penjumlahan.

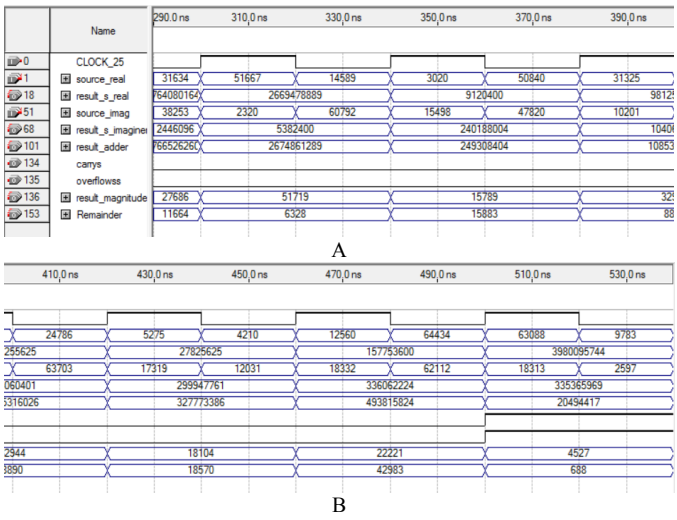
Operasi Penjumlahan		
Data Masukkan 1	Data Masukkan 2	Data Keluaran
2669478889	5382400	2674861289
9120400	240188004	249308404
981255625	104060401	1085316026
27825625	299947761	327773386
157753600	336062224	493815824
3980095744	335365969	4315461713

Dari hasil pengujian tersebut 5 data yang dihasilkan sesuai dengan hasil yang dilakukan secara manual, namun pada data ke-6 operasi pada blok komponen adder mengalami perbedaan hasil keluaran. Hal tersebut dikarenakan komponen adder beroperasi per satu bit, sehingga apabila operasi menghasilkan lebar bit yang lebih besar dari kapasitas komponen adder maka data yang akan digunakan adalah data yang memenuhi kapasitas komponen adder. Dari hasil pengujian ini, dapat dikatakan operasi telah memenuhi syarat operasi penjumlahan, namun apabila data melebihi kapasitas bit komponen maka data yang akan diambil adalah data yang memenuhi kapasitas bit komponen. Hal ini sebenarnya akan menjadikan pengaruh pada perangkat Audio Spectrum Analyzer, karena data yang seharusnya bernilai besar menjadi lebih kecil. Namun hal ini tidak menjadi pengaruh yang signifikan, sebab pengambilan data untuk tampilan, nantinya akan dipotong beberapa bit dan diambil nilai MSB-nya.

- **Tahap Akhir (*square root*)**

Pengujian terhadap blok komponen square root pada dasarnya sama dengan pengujian sebelumnya yakni data

keluaran akan dibandingkan dengan data perhitungan secara manual.



Gambar 9. (A) dan (B) Pengujian Kompoenen *Square Root.*

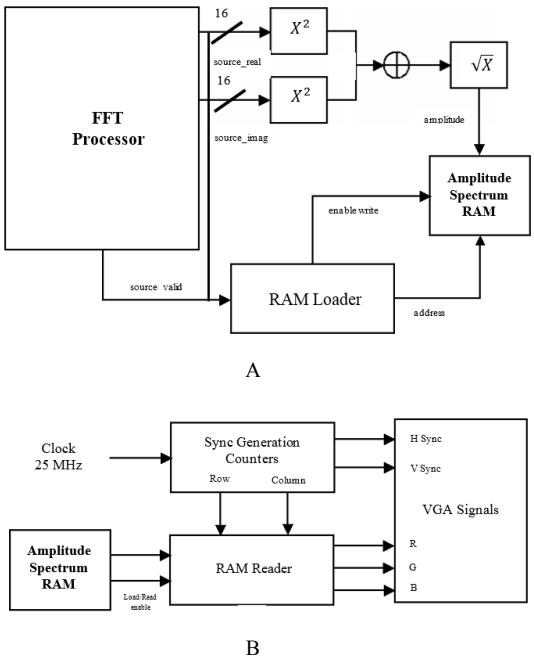
Tabel 6. Hasil perhitungan manual operasi akar kuadrat.

Operasi Akar Kuadrat	
Data Masukkan	Data Keluaran
2674861289	51719,06
249308404	15789,5
1085316026	32944,13
327773386	18104,51
493815824	22221,97
20494417	4527,076
(data ke-6) 4315461713	**65692,17**

Dari hasil pengujian data masukkan berupa data 32 bit mengalami pemecahan data menjadi 16 bit sebagai keluaran operasi dan 17 bit sebagai remainder. Penentuan pemecahan data ini telah dikonfigurasikan oleh blok komponen square root secara otomatis. Operasi dengan menggunakan blok komponen ini berbasis unsigned integer, yakni data yang diperoleh adalah data bilangan bulat. Sehingga jika hasil operasi square root dibandingkan dengan operasi secara manual, nilai data bilangan depan koma akan sama dengan nilai operasi menggunakan komponen square root. Nilai keluaran remainder mengartikan nilai sisa bilangan hasil operasi, atau dengan kata lain nilai ini adalah nilai belakang koma dari operasi secara manual. Terlihat pada data ke-6 apabila menggunakan operasi manual didapatkan nilai 65692.17, dengan melihat hasil ini apabila diterapkan pada bilangan 16 bit nilai ini melebihi kapasitas, dan tidak akan memenuhi persyaratan untuk ditampilkan di tahap selanjutnya.

2. Pengujian Memori *Magnitude*

Dari Gbr. 10 (A), data hasil operasi FFT Processor akan diproses bersamaan dengan kontroler RAM, sinyal yang mengindikasikan agar kontroler RAM bekerja adalah sinyal source_valid hasil keluaran FFT.



Gambar 10. (A) Proses data disimpan pada RAM. (B) Proses data dikirimkan ke VGA.

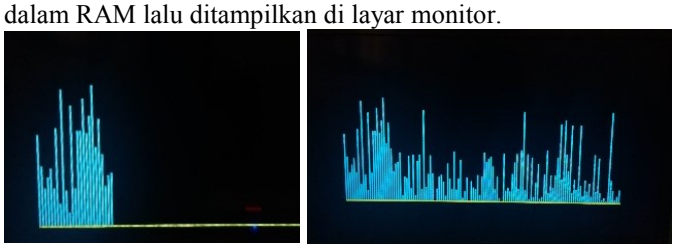
Ketika sinyal tersebut bernilai 1 maka akan mengaktifkan blok rangkaian persamaan *magnitude* dan blok kontroler RAM untuk bekerja. Data hasil operasi FFT berjumlah 256 data, namun data yang disimpan dalam RAM berjumlah 128 data melalui proses diagram blok (Gbr. 10. (B)).
Hal ini ditunjukkan pada kedua source code berikut :

Algoritma 1. *Source code counter* pada RAM.

```
process (clock, enable)
begin
    if enable = '0' then
        counter <= ( others => '0');
    elsif clock'event and clock = '1' then
        if  write_enable_signal = '1' then
            if counter = 127 then
                counter <= ( others => '0');
            else
                counter <= counter + 1;
            end if;
        end if;
    end if;
end process;

done_signal <= '1' when counter = 127 else
               '0';
```

Dari algoritma diatas, proses data untuk disimpan dalam RAM akan dihitung dari 0 hingga 127 atau dengan kata lain 128 data. Hal ini digunakan karena pada sisa dari data yakni dari data 128 hingga 255 merupakan cerminan dari data 0 hingga 127. Keadaan ini disebabkan karena sifat alami dari algoritma persamaan FFT. Pengujian dilakukan menggunakan

data masukkan bernilai acak, yang kemudian disimpan di dalam RAM lalu ditampilkan di layar monitor.

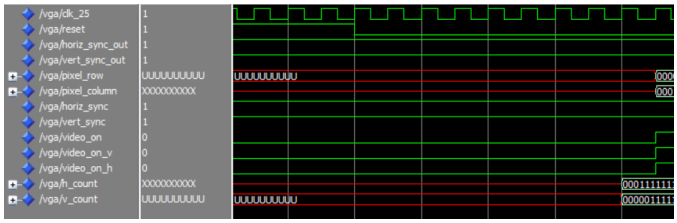


Gambar 11. (A) Tampilan VGA 25 Data Acak. (B) Tampilan VGA 128 Data Acak.

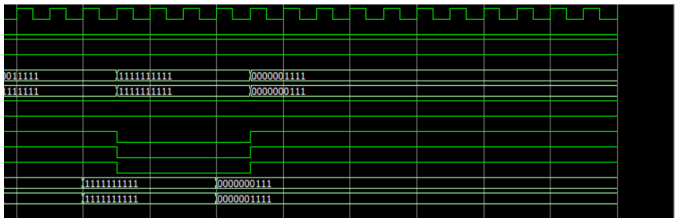
Pada Gbr. 11. (A) menggunakan data masukkan berjumlah 25 data, dan pada Gbr. 11. (B) menggunakan data masukkan berjumlah 128 data. Dari hasil pengujian tersebut dikatakan bahwa blok komponen kontroler RAM dan komponen RAM bekerja sesuai dengan yang diinginkan.

3. Pengujian Tampilan VGA Monitor

Pengujian terhadap tampilan VGA monitor dilakukan melalui 2 tahap yakni pengujian terhadap komponen sinkronisasi dan pengujian terhadap komponen tampilan grafis.



A



B

Gambar 12. (A) dan (B) Pengujian terhadap komponen sinkronisasi

Gbr. 12. (A) dan (B) merupakan gambar hasil pengujian terhadap komponen sinkronisasi. Gbr. 12. (A) dan (B) tersebut membuktikan bahwa keluaran berupa mapping pada baris dan kolom ketika *counter* menghitung data masukkan yang diterima. Hal ini bersesuaian dengan yang diinginkan, sebab data masukkan berupa data akan diverifikasi terlebih dahulu melalui *counter* sehingga tampilan yang diinginkan bersesuaian.

Pengujian terhadap tampilan grafis terbagi menjadi 2 bagian yakni pengujian terhadap tampilan karakter dan pengujian terhadap tampilan histogram.

Pengujian terhadap tampilan karakter dilakukan dengan cara pemanggilan data pada *Read Only Memory* (ROM) yang telah dirancang membentuk suatu karakter. Perancangan karakter dilakukan menggunakan penyimpanan *file* dalam bentuk *memory initialization file* (mif). Pemanggilan ROM dilakukan menggunakan source code berikut :

Algoritma 2. *Source code* pemanggilan karakter dari ROM.

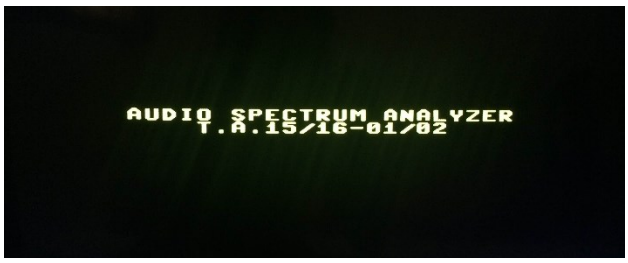
```
process
    begin
    wait until (clock25 'event) and (clock25 = '1');
        case V_Counter(9 downto 4) is
            when "000011"  =>
                case posisi_pixel_X_buf(9 downto 4) is

                    when "001000" => alamat_karakter_1 <= "000000"; -- A
                    when "001001" => alamat_karakter_1 <= "000011"; -- U
                    when "001010" => alamat_karakter_1 <= "000100"; -- D
                    when "001011" => alamat_karakter_1 <= "000101"; -- I
                    when "001100" => alamat_karakter_1 <= "000110"; -- O

                    when "001110" => alamat_karakter_1 <= "000010"; -- S
                    when "001111" => alamat_karakter_1 <= "000111"; -- P
                    when "010000" => alamat_karakter_1 <= "001000"; -- E
                    when "010001" => alamat_karakter_1 <= "001001"; -- C
                    when "010010" => alamat_karakter_1 <= "001010"; -- T
                    when "010011" => alamat_karakter_1 <= "001011"; -- R
                    when "010100" => alamat_karakter_1 <= "000011"; -- U
                    when "010101" => alamat_karakter_1 <= "001100"; -- M

                    when "010111" => alamat_karakter_1 <= "000000"; -- A
                    when "011000" => alamat_karakter_1 <= "001101"; -- N
                    when "011001" => alamat_karakter_1 <= "000000"; -- A
                    when "011010" => alamat_karakter_1 <= "001110"; -- L
                    when "011011" => alamat_karakter_1 <= "001111"; -- Y
                    when "011100" => alamat_karakter_1 <= "010000"; -- Z
                    when "011101" => alamat_karakter_1 <= "001000"; -- E
                    when "011110" => alamat_karakter_1 <= "001011"; -- R

                    when others    => alamat_karakter_1 <= "000001" ; -- Space
                end case;
```

Dari *source code* pemanggilan data pada ROM kemudian disinkronisasi untuk ditampilkan pada layar menggunakan sinyal warna sebagai berikut :

Algoritma 3. *Source code* pewarnaan tampilan karakter.

```
    elsif rom_mux_output = '1' then
    VGA_R <= "1111111111"; VGA_G <= "1111111111"; VGA_B <= "0000000000";
```

Pengujian dilakukan secara langsung pada VGA Monitor sebagai berikut :

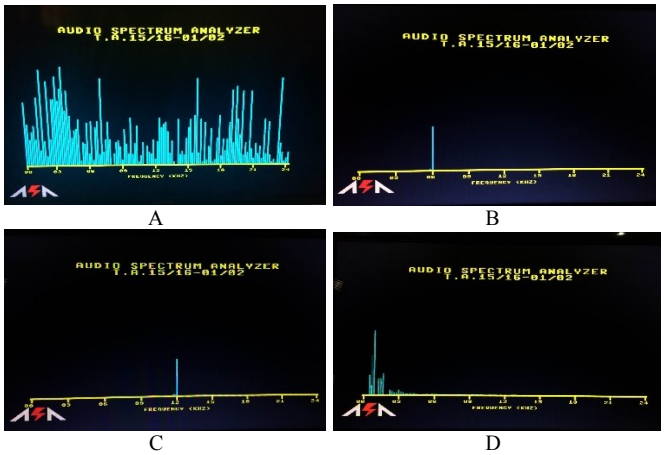


Gambar 13. Pengujian terhadap penampilan karakter pada VGA

Pada Gbr. 13. terlihat bahwa tampilan karakter muncul pada VGA Monitor, hal ini menunjukkan bahwa pengujian terhadap tampilan karakter pada VGA Monitor telah sesuai dengan yang diharapkan.

Pada pengujian terhadap tampilan histogram menerapkan spesifikasi dari frekuensi bin yang digunakan yakni 187,5 Hz. Hal ini menunjukkan bahwa satu nilai histogram diwakilkan 187,5 Hz. Pengujian terhadap tampilan histogram dilakukan

menggunakan 2 data masukkan yang pertama menggunakan data masukkan (*dummy data*) secara acak yang kedua menggunakan perangkat *Audio Spectrum Analyzer* berbasis FPGA melalui data masukkan sinyal tertentu dan suara. Pengujian dilakukan untuk memperlihatkan apakah data masukkan sesuai dengan data keluaran yang ditampilkan pada VGA Monitor.



Gambar 14. (A) Pengujian tampilan VGA menggunakan 128 data acak. (B) Pengujian tampilan VGA menggunakan masukkan 6 KHz. (C) Pengujian tampilan VGA menggunakan masukkan 12 KHz. (D) Pengujian tampilan VGA menggunakan masukkan suara (musik).

Terlihat pada Gbr. 14. (A) menunjukkan bahwa data masukkan sesuai dengan data keluaran, yang mana nilai dimasukkan secara acak sejumlah 128 data, dan dalam tampilan ditunjukkan data sejumlah 128 dengan keakuratan sesuai dengan penempatan data tersebut. Pada Gbr 14. (B) dan (C) menunjukkan bahwa data masukkan menggunakan sinyal 6 KHz dan 12 KHz, pada tampilan VGA Monitor, ditampilkan nilai spektrum yang sesuai yakni pada 6 KHz dan 12 KHz sehingga menunjukkan keakuratan data masukkan dan data keluaran. Pada Gbr. 14. (D) menunjukkan bahwa data masukkan berupa suara musik melalui kabel *auxiliary* (AUX). Pada Gbr. 14. (D) tampilan histogram berada di rentang 20 Hz – 3 KHz hal ini menunjukkan bahwa pada rentang frekuensi musik ataupun nada berada pada frekuensi 20 Hz – 3 KHz.

V. KESIMPULAN

Melihat pembahasan yang telah dijabarkan dari bagian perancangan hingga pengujian, dapat ditarik kesimpulan bahwa Penerapan sistem persamaan *magnitude* dapat diterapkan pada blok komponen sehingga mampu diintegrasikan dengan perangkat *Audio Spectrum Analyzer* Berbasis FPGA. Kendali pada memori *magnitude* mampu memotong 256 data menjadi 128 data, hal ini dilakukan karena sisa data merupakan cerminan dari 128 data sebelumnya. Kondisi ini terjadi dikarenakan sifat natural dari alogritma FFT dengan masukkan bilangan real. Tampilan spektrum dalam bentuk histogram mampu ditampilkan pada VGA monitor. Data yang tersimpan pada memori magnitude mampu diakses oleh blok VGA yang kemudian ditampilkan pada VGA monitor sesuai dengan data yang diperoleh dari memori *magnitude*. Sistem persamaan *magnitude*, memori *magnitude*, dan tampilan VGA mampu diintegrasikan sesuai dengan spesifikasi sistem. Keseluruhan sistem (persamaan *magnitude*, memori *magnitude*, dan tampilan VGA) mampu diintegrasikan dengan perangkat *Audio Spectrum Analyzer* Berbasis FPGA. Untuk pengembangan masa depan diaharapkan dilakukan pengembangan terhadap komponen adder pada blok komponen persamaan magnitude, untuk memperoleh data yang utuh, nilai data yang melebihi kapasitas bit dapat digunakan pula sehingga data yang diperoleh untuk operasi selanjutnya lebih presisi. Kemudian pengembangan terhadap penambahan sampel sinyal audio dan peningkatan resolusi tampilan VGA sehingga mampu meningkatkan keakuratan data dan kejelasan terhadap proses analisis.

VI. REFERENSI

[1] Elektronika Dasar. Sinyal Audio (Gelombang Suara) [online]. Tersedia : elektronika-dasar.web.id/sinyal-audio-gelombang-suara/

[2] Wibowo, Ferry Wahyu. 2014. “FPGA dan VHDL Teori, Antarmuka dan Aplikasi”. Deepublish. Sleman.

[3] Kristianti, Veronica Ernita. Pemrograman Devais FPGA [online]. Tersedia : veronica.staff.gunadarma.ac.id/ Downloads/files/32911/Pemrograman+Devais+FPGA+Pert.1%262.pptx

[4] Pong P. Chu. 2008. FPGA Protoryping By VHDL Examples XILINX SPARTAN-3 VERSION. Cleveland State University. 2008

[5] Enoch Hwang (2004). “Build a VGA Monitor Controller”. Circuit Cellar. November 2004

[6] Laboratorium Dasar Teknik Elektro. 2013. Praktikum Sistem Digital. Sekolah Teknik Elektro dan Informatika-Institut Teknologi Bandung. 2013

[7] James Darabi & Tracy Cline. (2013). Implementation of a Real-Time Spectrum analyzer, Research Document, Indiana University, October 2013.

[8] Don Lim Yuan Heng. 2011.”Design a voice recorder using FPGA”. Project Report, SIM University, January 2011.

[9] TinyVGA.com. “VGA Signal 640 x 480 @60 Hz Industry Standart timing” [online]. Tersedia : tinyvga.com/vga-timing/640x480@60Hz

[10] Min-Chuan, Lin. 2006. An Approach to FPGA-based Time-Frequency Spectrogram by Real-Time Sweep Spectral Extraction Algorithm. Department of Electronics Engineering, Kun-Shan University, Taiwan.

[11] Jurgen Schuhmacher. 2011. Audio Spectrum analyzer with Cyclone IV. Tersedia: http://www.96khz.org/htm/spectrumanalyzer2.htm.

[12] Tectonics (2004), “Fundamentals of Real-Time Spectrum Analysis”, Technical report, Tektronics, 2004.

[13] Mesbah Uddin Mohammed. 2009. Audio Spectrum Analyzer. Ajman University Of Science & Technology.

[13] Agilent(2004), “Swept and FFT Analysis”. Performance spectrum analyzer series – application note, Agilent, 2004.

[14] Shimshon Levy. 2012. “Spectrum analyzer and Spectrum Analysis”. October 2012

[15] Barbaro Maykel. 2013. FPGA based spectrum analyzer. Conference Paper, University of Buenos Aires. August 2013.