Makalah Seminar Tugas Akhir

IMPLEMENTASI ALGORITMA FFT (*FAST FOURIER TRANSFORM*) PADA *DIGITAL SIGNAL PROCESSOR* (DSP) TMS320C542

Nandra Pradipta, L2F000623
Jurusan Teknik Elektro Fakultas Teknik Universitas Diponegoro

Abstrak- *Pengolahan sinyal digital sekarang diaplikasikan pada banyak bidang seperti* biomedical, *navigasi, telekomunikasi, pengolahan suara dan musik, serta pengolahan video dan gambar. Transformasi fourier diskrit (DFT) memainkan peranan penting dalam banyak aplikasi pengolahan sinyal digital, termasuk pentapisan linier, analisis korelasi, dan analisis spektrum.*

Dalam Tugas Akhir ini akan dibuat algoritma FFT dengan menggunakan Digital Signal Processor *TMS320C542.* Fast Fourier Transform *(FFT) adalah suatu algoritma untuk menghitung* Discrete Fourier Transform *(DFT) yang digunakan untuk menghitung spektrum frekuensi sinyal yang telah dicuplik komputer dan* FFT *merupakan prosedur penghitungan DFT yang efisien sehingga akan mempercepat proses penghitungan DFT yang secara substansial dapat lebih menghemat waktu dari pada metoda yang konvensional. Metode FFT yang digunakan adalah algoritma FFT radiks dua dengan* decimation in time.

Pengujian dilakukan dengan variasi sinyal berupa sinus, segitiga dan kotak dengan variasi frekuensi dan jumlah cuplikan. Sedangkan frekuensi cuplikan telah ditentukan sebelumnya sebesar 8000Hz.

Kata kunci: fast Fourier transform, discrete Fourier transform, decimation in time

I. PENDAHULUAN

1.1 Latar Belakang Masalah

DSP atau *Digital Signal Processor* merupakan salah satu tipe dari mikroprosesor yang mempunyai kemampuan yang relatif besar dengan kecepatan yang relatif tinggi. DSP digunakan di berbagai produk elektronika, misalnya telepon selular dan modem.

Salah satu aplikasi dari DSP adalah untuk membuat algoritma Transformasi Fourier Cepat atau *Fast Fourier Transform* (FFT). Transformasi Fourier Cepat (FFT) merupakan metode perhitungan DFT (*Dicrete Fourier Transform*) yang sangat efisien sehingga akan mempercepat proses perhitungan DFT. Kecepatannya berasal dari kenyataan bahwa algoritma FFT memanfaatkan hasil-hasil komputasi sebelumnya untuk mengurangi banyaknya operasi. Khususnya, algoritma FFT memanfaatkan sifat keperiodikan dan sifat simetri fungsi-fungsi trigonometri pada runtun eksponensial kompleks untuk menghitung transformasi tersebut dengan sekitar $N\log_2 N$ operasi, berbeda dengan menggunakan metode perhitungan DFT yang mencapai N^2 operasi.

1.2 Tujuan

Tujuan dalam Tugas Akhir ini adalah mengimplementasikan algoritma Transformasi Fourier Cepat (FFT) pada *Digital Signal Processor* TMS320C542 menggunakan DSP S*tarter Kit*, DSKplus.

1.3 Pembatasan Masalah

Dalam tugas Akhir ini ada beberapa pembatasan masalah sebagai berikut :

1. Algoritma FFT yang digunakan adalah radiks dua dengan metode *Decimation in Time*.
2. Masukan berupa sinyal analog yang dicuplik diasumsikan bilangan riil semua.
3. Sinyal yang diujikan adalah sinyal sinus, segitiga dan kotak dengan frekuensi 500Hz, 1000Hz dan 2000Hz.
4. Frekuensi cuplik ditetapkan 8000 Hz.
5. Pengujian dilakukan hanya untuk mendapatkan tanggapan frekuensi.

II. DASAR TEORI

Sebuah sinyal dapat ditinjau dari dua sudut yang berbeda yaitu kawasan waktu dan kawasan frekuensi. Sinyal pada kawasan waktu dapat diubah menjadi sinyal dalam kawasan frekuensi demikian juga sebaliknya. Kedua kawasan ini mempunyai informasi yang saling melengkapi dari data yang sama.

2.1 Alihragam Fourier Diskret

Alihragam Fourier Diskret atau *Discrete Fourier Transform* (DFT) digunakan untuk menghitung spektrum frekuensi sinyal pada komputer digital. Alihragam Fourier diskret ditunjukkan dalam Persamaan (2.1) berikut.

$$H(k) = \sum_{n=0}^{N-1} h(n) e^{-j2\pi nk/N} \quad \text{untuk } 0 \le k \le (N-1) \qquad (2.1)$$

dengan $h(n)$ adalah runtun masukan diskret dan $H(k)$ merupakan *magnitude* frekuensi serta N merupakan jumlah runtun masukan diskret.

Alihragam Fourier diskret balik atau *Inverse Discrete Fourier Transform* (IDFT) ditentukan dengan cara menghitung runtun waktu diskret $h(n)$ dari runtun frekuensi diskret $H(k)$.

$$h(n) = \frac{1}{N} \sum_{k=0}^{N-1} H(k) e^{j2\pi nk/N} \quad ,0 \le n \le (N-1) \qquad (2.2)$$

2.2 Algoritma Alihragam Fourier Cepat

Algoritma Alihragam Fourier Cepat atau *Fast Fourier Transform* (FFT) merupakan prosedur penghitungan DFT yang efisien sehingga akan mempercepat proses penghitungan DFT. Bila diterapkan pada kawasan waktu maka algoritma ini disebut juga sebagai FFT penipisan dalam waktu atau *decimation-in-time* (DIT). Penipisan kemudian mengarah pada pengurangan yang signifikan dalam sejumlah perhitungan yang dilakukan pada data kawasan waktu.

Persamaan (2.1) menjadi

$$H_1(k) = \sum_{n=0}^{N-1} h(n) W_N^{nk} \text{ , untuk } 0 \le k \le (N-1) \qquad (2.3)$$

dimana faktor $e^{-j2\pi/N}$ akan ditulis sebagai ,

$$W_N = e^{-j2\pi/N} = \cos(2\pi/N) - j\sin(2\pi/N) \quad (2.4)$$

Akhiran *n* pada Persamaan (2.3) diperluas dari *n*=0 sampai dengan *n*=*N*-1, bersesuaian dengan nilai data *h*(0), *h*(1), *h*(2), *h*(3)...*h*(*N*-1). Runtun bernomor genap adalah *h*(0), *h*(2), *h*(4)....*h*(*N*-2) dan runtun bernomor ganjil adalah h(1), *h*(3)....*h*(*N*-1). Kedua runtun berisi *N*/2-titik. Runtun genap dapat ditandakan *h*(2*n*) dengan *n*=0 sampai *n*=*N*/2-1, sedangkan runtun ganjil menjadi *h*(2*n*-1). Kemudian Persamaan (2.3) dapat ditulis ulang menjadi

$$H_1(k) = \sum_{n=0}^{N/2-1} h(2n) W_N^{2nk} + \sum_{n=0}^{N/2-1} h(2n+1) W_N^{2(2n+1)k}$$

$$= \sum_{n=0}^{N/2-1} h(2n) W_N^{2nk} + W_N^k \sum_{n=0}^{N/2-1} h(2n+1) W_N^{2nk},\ 0 \le k \le (N-1) \quad (2.5)$$

Selanjutnya dengan menggantikan $W_N^{2nk} = W_{N/2}^{nk}$, maka Persamaan (2.5) menjadi

$$H(k) = \sum_{n=0}^{N/2-1} h(2n) W_{N/2}^{nk} + W_N^k \sum_{n=0}^{N/2-1} h(2n+1) W_{N/2}^{nk} \quad (2.6)$$

2.4 Pembalikan Bit (*Bit-Reversal*)

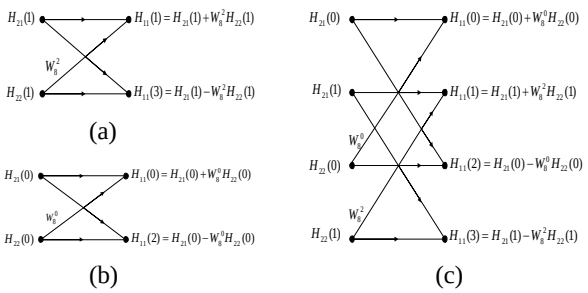
Algoritma d*ecimation in time* mengharuskan masukan disusun ulang sedemikian rupa sehingga hasil perhitungan akhir mempunyai urutan yang sesuai. Tabel 2.1 menunjukkan bagaimana suatu masukan disusun ulang untuk tujuan tersebut.

Tabel 2.1 Pembalikan bit.

Masukan yang Asli		Masukan yang Berkebalikan Bit	
Desimal	Biner	Biner	Desimal
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

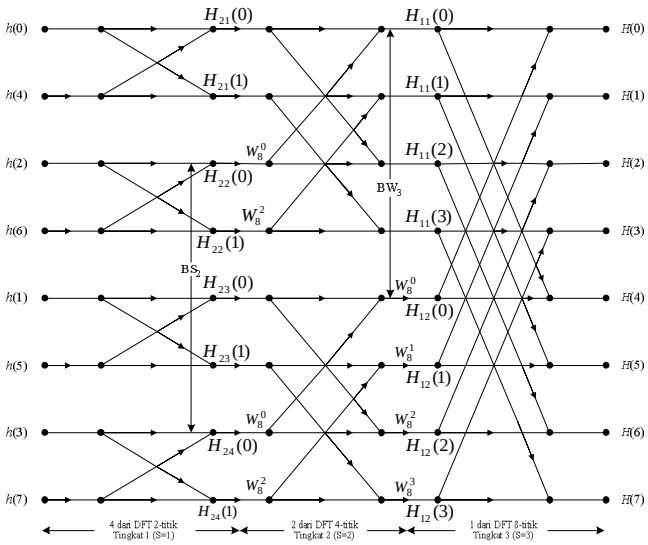
2.5 Algoritma Kupu-Kupu (*Butterfly*)

Penjelasan pada subbab 2.2 dapat direpresentasikan dalam bentuk diagram dengan memanfaatkan sifat simetri yang dipusatkan pada perbedaan tanda dan mengambil persamaan dalam pasangan. Hal ini diilustrasikan dalam Gambar 2.1a. Masukan berada pada sebelah kiri gambar silang dan keluarannya berada di sebelah kanan. Gambar 2.1b menunjukkan bagaimana $H_{11}(1)$ dan $H_{11}(3)$ didapat dari diagram. Dengan *overlapping* Gambar 2.1a dan 2.1b didapatkan diagram gabungan dimana keluaran DFT disusun untuk meningkatkan *k*. Hal ini ditunjukkan dalam Gambar 2.1c. Struktur pada Gambar 2.1a dikenal dengan *butterfly*. 8-titik FFT digambarkan seperti dalam Gambar 2.2.



Gambar 2.1 Struktur algoritma *butterfly* pada FFT.

Berikut ini adalah contoh perhitungan untuk mendapatkan nilai DFT dari runtun data {1,0,0,1). Perhitungan tersebut menggunakan *decimation-in-time* yang terlihat pada Gambar 2.2 yang merupakan perhitungan DFT 4-titik dengan *h*(0)=1, *h*(1)=0, *h*(2)=0, *h*(3)=1 dan $H_1(k) = H_{11}(k) + W_N^k H_{12}(k)$. Runtun yang disusun ulang adalah *h*(0), *h*(2), *h*(1), *h*(3).



Gambar 2.2 Algoritma *butterfly* FFT untuk DFT 8-titik.

Nilai-nilai hasil perhitungan ini akan sama dengan yang didapat bila menggunakan perhitungan DFT pada sub bab 2.3, tetapi proses perhitungannya lebih cepat menggunakan algoritma FFT. Kesimpulan ini adalah umum dan penghematan perhitungan semakin meningkat jika jumlah data meningkat.

Keuntungan perhitungan pada FFT dapat diilustrasikan dengan memperhatikan algoritma FFT pada Gambar 2.2. Gambar ini menunjukkan bahwa sebuah FFT *N*-titik berisi *N*/2 kupu-kupu per tahap dan $\text{Log}_2 N$ tahap sehingga total berisi $(N/2)\text{Log}_2 N$ kupu-kupu. Gambar 2.1a menunjukkan bahwa masing-masing kupu-kupu mengandung satu perkalian kompleks dari bentuk $W_N^k H_{ij}(k)$. Oleh karena itu FFT mengandung $(N/2)\text{Log}_2 N$ perkalian kompleks dibandingkan dengan N^2 pada DFT. Penghematan perhitungan dalam perkalian kompleks adalah $N^2 - (N/2)\text{Log}_2 N$. Masing-masing kupu-kupu berisi dua penambahan kompleks sehingga FFT membutuhkan $N\text{log}_2 N$ dibandingkan dengan $N(N-1)$ untuk DFT. Penghematan penambahan kompleks adalah $N(N-1) - N\text{log}_2 N$. Penghematan ini diilustrasikan dalam Tabel 2.2.

Tabel 2.2 Perbandingan perhitungan kompleks pada DFT dan FFT.

$N\text{log}_2$	*N*	DFT		FFT	
		Perkalian Kompleks	Penambahan Kompleks	Perkalian Kompleks	Penambahan Kompleks
2	4	16	12	4	8
3	8	64	56	12	24
4	16	256	240	32	64
5	32	1 024	992	80	160
6	64	4 096	4 032	192	384
7	128	16 384	16 256	448	896
8	256	65 536	65 280	1 024	2 048
9	512	262 144	261 632	2 304	4 608
10	1024	1 048 576	1 047 552	5 120	10 240

2.6 Tahap-tahap Perancangan Algoritma FFT

Perancangan algoritma FFT meliputi empat langkah, yaitu:

a. Penyusunan runtun masukan menggunakan pembalikan bit.
b. Perhitungan FFT kompleks *N*-titik menggunakan metoda kupu-kupu.
c. Pemisahan bagian genap dan ganjil.
d. Perhitungan keluaran akhir.

2.7 *Digital Signal Processor*

Kebanyakan prosesor yang ada sekarang ini berdasarkan pada konsep Von Neumann. Pada operasi *real-time*, sebuah prosesor DSP harus memiliki arsitektur tersendiri yang dioptimalkan untuk menjalankan fungsi pengolahan sinyal, yang ditandai oleh:

1. Struktur bus jamak dengan ruang memori yang terpisah untuk data dan untuk program.
2. Port I/O untuk melewatkan data ke dan dari piranti eksternal seperti ADC dan DAC atau untuk melewatkan data digital ke prosesor lainnya.
3. Unit aritmatika untuk operasi aritmetika dan logika, yang terdapat dalam ALU, dan juga perangkat keras pengali (*hardware multiplier*).
4. MAC (M*ultiplier-Accumulator*) adalah dasar untuk DSP. Karena semua sistem fisik dapat dimodelkan dengan deret Taylor, maka kemampuan untuk mengalikan dua bilangan dan dijumlahkan dengan hasil sebelumnya dapat menjadi suatu keuntungan

Arsitektur tersebut diperlukan dalam sebuah prosesor DSP karena kebanyakan algoritma DSP (seperti penapisan, korelasi dan FFT) memerlukan operasi aritmetika yang berulang-ulang seperti perkalian, penjumlahan, akses memori dan aliran data yang padat yang melalui CPU. Arsitektur dari mikroprosesor yang standar tidak cocok untuk jenis aktifitas ini. Dalam *Digital Signal Processor*, beberapa teknik berikut digunakan:

1. Arsitektur Harvard.
2. *Pipelining.*
3. *Hardware Multiplier-accumulator*.
4. Instruksi-instruksi khusus.
5. Memori/*chace on-chip*.

2.3 DSKplus TMS320C542

DSKplus TMS320C542 adalah DSP *starter kit* dengan arsitektur yang lebih dikembangkan. Alat pengembangan ini berupa *board* aplikasi yang dihubungkan dengan PC sehingga dapat digunakan untuk menjelajahi arsitektur dan operasi dari CPU dan *peripheral* TMS320C542. *Board* DSKplus mengeksekusi kode program TMS320C542 dalam *real time*, di dalam PC terdapat *debugger* berbasis Windows yang dapat menganalisis kode tersebut baris demi baris, menampilkan informasi register internal DSP, dalam *real time* juga. Fitur-fitur yang terdapat dalam DSKplus antara lain adalah:

a. DSP TMS320C542 *fixed-point*.
b. 40 MIPS (siklus waktu instruksi 25 ns).
c. 10K *word dual-access* RAM (DARAM).
d. 2K *word* boot ROM.
e. Satu port serial *time division multiplexed* (TDM).
f. Satu port serial tersangga (*buffered serial port*).
g. Satu *host port interface* (HPI) untuk komunikasi PC ke DSP.
h. Satu *timer on-chips*.
i. Rangkaian antarmuka ADC, DAC TLC320AC01 yang dapat diprogram dengan kualitas suara.
j. *Header* emulator XDS510.
k. Bus ekspansi I/O dan sinyal kontrol untuk perancangan eksternal.
l. Mini jack mono 1/8 inchi untuk I/O analog.

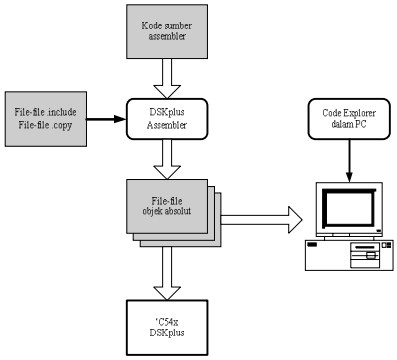
Antara DSKplus dan host terdapat protokol komunikasi bersama. PC host melakukan *download* protokol ke perangkat lunak komunikasi DSP, yang menempati memori DSP pada alamat 80h – 17Fh. Protokol juga menggunakan buffer komunikasi bersama di dalam memori DSP pada 1000h –1009h. Semua memori dengan lokasi dari 80h – 17Fh dan 1000h – 1009h dicadangkan dan tidak boleh ditulis ulang. DSKplus menyediakan enam slot untuk *header* yang dapat digunakan untuk keperluan ekspansi *board*. Dari keenam port tersebut terdapat slot *header* untuk bus data, bus alamat, dan bus kontrol yang dapat digunakan untuk menambahkan memori atau *peripheral* lainnya ke TMS320C542.

Osilator *on-board* 10-MHz menyediakan detak (*clock*) ke TMS320C542 , ‘AC01 dan perangkat PAL. Pilihan PLL pada TMS320C542 diset 4 yang menghasilkan detak internal 40 MHz, sedangkan ‘AC01 berjalan pada detak 10 MHz.

III. IMPLEMENTASI TAPIS DIGITAL

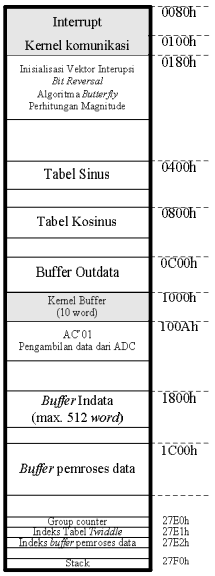
Implementasi algoritma FFT pada DSP TMS320C542 adalah menentukan terlebih dahulu frekuensi cuplik (f_s) dan jumlah cuplikan ($2N$) yang akan dihitung. Masukan berasal dari pembangkit frekuensi audio (AFG, *Audio Frequency Generator*), sinyal analog dari AFG ini diubah ke dalam sinyal digital oleh ADC AC01 dengan resolusi 14 bit. Cuplikan-cuplikan yang akan dihitung disimpan dalam memori. Setelah nilai cuplikan didapatkan kemudian dilakukan perhitungan dengan algoritma FFT. Hasil perhitungan disimpan dalam memori kemudian diplot dalam grafik magnitude kawasan frekuensi. Sinyal analog yang akan dihitung antara lain sinyal sinus, segitiga dan kotak dengan variasi nilai frekuensi dan variasi jumlah cuplikan. Perhitungan dilakukan pada bentuk sinyal dan frekuensi yang sama dengan jumlah cuplikan yang berbeda-beda dan dibandingkan hasil masing-masing.

Program terdiri dari inisialisasi ADC, inisialisasi vektor intrupsi, tabel nilai-nilai sinus dan kosinus (*twiddle*), algoritma FFT yang meliputi pembalikan bit (*bit reversal*), algoritma kupu-kupu (*butterfly*) dan perhitungan magnitude. Hasilnya disimpan dalam memori, dan pengamatan dilakukan dengan fasilitas grafik pada *debugger* atau dibuat grafik magnitudenya dengan Microsoft Excel.



Gambar 3.1 Alir pengembangan perangkat lunak DSKplus *Assembler*

Adapun alokasi memori file tersebut adalah sebagai berikut:



Gambar 3.2 Alokasi memori.

3.1 Inisialisasi Vektor Interupsi

Vektor interupsi dapat dipetakan ulang pada awal dari sembarang 128-*wordpage* pada program memori kecuali daerah yang dicadangkan. Pada saat reset bit-bit pada IPTR diset ke 1 (IPTR = 1FFh). Vektor interupsi dapat dipetakan ke lokasi lain dengan mengisi IPTR dengan nilai selain 1FFh. Sebagai contoh, vektor interupsi dapat dipindahkan untuk mulai pada lokasi 0080h dengan mengisi IPTR dengan 0001h. Vektor *hardware reset* ($\overline{RS}$) tidak dapat dipetakan ulang karena *hardware reset* mengisi IPTR dengan 1. Oleh karena itu, vektor reset untuk *hardware reset* selalu mengambil lokasi FF80h pada ruang program.

Ketika *debugger* digunakan, lokasi vektor reset harus diubah disesuaikan dengan *kernel*. DSKplus debugger menggunakan empat vektor interupsi, yaitu $\overline{RESET}$, TRAP2, $\overline{TRINT}$, dan $\overline{HPINT}$. Semua lokasi vektor yang lain bebas digunakan. Jika akan menggunakan *debugger*, keempat vektor interupsi di atas tidak boleh diubah-ubah.

Bit-bit IPTR merupakan bagian dari PMST (*Processor Mode Status Register*), sehingga untuk menetapkan nilai IPTR, nilai PMST yang harus ditetapkan.

3.2 Inisialisasi AIC (*Analog Interface Circuit*)

Pada *board* DSKplus terdapat ADC/DAC yaitu TLC320AC01C Single-Supply Analog Interface Circuit. AIC ini mempunyai sembilan register data. Register yang digunakan adalah register 1 (register A), register 2 (register B), dan register 4. Register lainnya dalam keadaan *default*. Frekuensi cuplik ditetapkan dengan rumus:

Frekuensi cuplik = $f_s = \dfrac{FCLK}{nilai\_register\_B}$

$$fs = \frac{f(MCLK)}{2(nilai\_register\_A)\times(nilai\_register\_B)}$$

Register 4 digunakan untuk mengatur perolehan tegangan (*gain*) input dan output.

3.3 Tabel Nilai-nilai Sinus dan Kosinus

Algoritma FFT di dalamnya terdapat perhitungan nilai-nilai sinus dan kosinus. Perhitungan nilai sinus atau kosinus menggunakan deret Maclaurin dalam pemrograman bahasa tingkat rendah (*assembly*).

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots$$

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots$$

Perhitungan ini sendiri diperlukan suatu fungsi tersendiri yang rumit, untuk memudahkan pemrograman, nilai-nilai sinus dan kosinus ini dibuat dalam bentuk tabel (*look up table*) yang disebut *twiddle*. Tabel sinus diberi nama *twiddle*1, sedangkan tabel kosinus *twiddle*2. Tabel ini berisi nilai-nilai sinus dan kosinus 0° sampai dengan 180° dengan selisih 0,352°, sehingga dalam satu tabel terdapat 512 variasi nilai sinus atau kosinus. Hal ini dikarenakan jumlah maksimum cuplikan (*N*) 512-titik.

Nilai-nilai sinus dan kosinus harus diubah dalam bentuk format Q15 agar bisa disimpan dalam memori. Pengubahan ini dilakukan dengan mengalikan hasil nilai sinus atau kosinus dengan 2^{15}. Misal:

$\sin 45° = 0.7071 \rightarrow \sin 45° = 0.7071 \times 2^{15} = 23170.4$
$\approx 23170 = 5A82h$

$\cos 22.5° = 0.9239 \rightarrow \cos 22.5° = 0.9239 \times 2^{15} = 30273.6$
$\approx 30274 = 7642h$

3.4 Pengambilan Data dari ADC

Data masukan pada tugas akhir ini berasal dari konversi sinyal analog menjadi data digital dan kemudian disimpan pada memori dengan alamat 1800h. Berikut adalah penggalan program yang akan menyimpan data dari ADC ke dalam buffer memori masukan pada alamat 1800h.

3.5 Program Utama Algoritma FFT

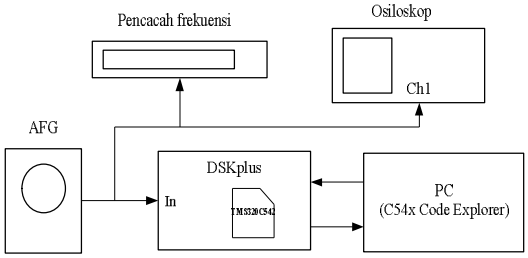
Program utama algoritma FFT terdiri dari proses pembalikan bit, algoritma kupu-kupu dan perhitungan magnitude. Program utama ini juga memanggil *file* inisialisasi berekstensi *.asm* yang telah dibuat yaitu inisialisasi vektor interupsi, inisialisasi AIC, dan inisialisasi koefisien *twiddle*. Program perhitungan FFT ini menggunakan algoritma FFT radiks-2 dengan metode DIT.

Setiap runtun masukan real dengan jumlah cuplikan 16 sampai 512 dapat digunakan dengan modifikasi nilai *N* yang terdapat pada *file* initfftx.asm. Sebagai contoh untuk masukan 256-titik *N* harus diset menjadi 128 dan *LogN* bernilai 7. Diagram alir program utama algoritma FFT ditunjukkan pada Gambar 3.3.

Gambar 3.3 Diagram alir program utama algoritma FFT

IV. PENGUJIAN DAN ANALISIS

Pengujian dilakukan untuk memperoleh hasil perhitungan FFT dengan memberikan sinyal uji pada input analog DSKplus TMS320C542 Stater Kit dari pembangkit frekuensi audio (AFG), yang secara bersamaan diamati dengan pencacah frekuensi (*frequency counter*) untuk mengetahui frekuensi yang dihasilkan dan osiloskop untuk melihat amplitude sinyal dan hasil perhitungannya disimpan pada memori DSP. Blok pengujian ini ditunjukkan pada Gambar 4.1.



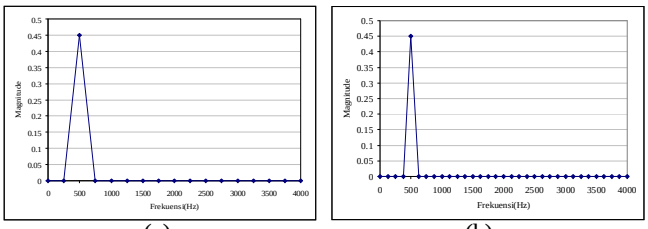
Gambar 4.1 Blok pengujian tapis digital.

Sinyal masukan dari AFG yang dicuplik, pada pengujian ini diproses dengan perhitungan FFT *N*-titik yang sesuai dengan banyaknya jumlah cuplikan tersebut. Hasil perhitungan yang berupa nilai riil dan imajiner diolah untuk memperoleh nilai amplitude dengan rumus pada Persamaan (4.1)

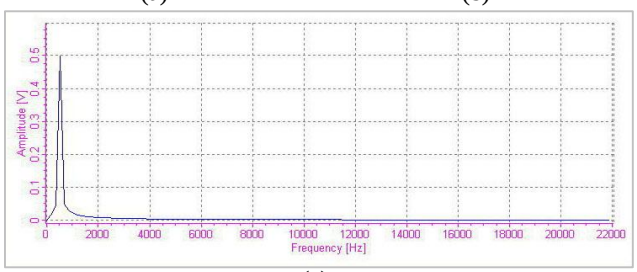
$$A(f)=\sqrt{\mathrm{Re}(f)^2+\mathrm{Im}(f)^2}$$

dimana f adalah besar frekeuensi pada amplitude yang akan dihitung. Hasil tanggapan frekuensi dibuat grafik hubungan frekuensi terhadap nilai amplitude yang diperoleh. Hasil pengujian tanggapan frekuensi ini juga akan dibandingkan dengan tanggapan frekuensi yang dihasilkan oleh program simulasi FFT Properties.

Hasil pengujian tapis ditunjukkan pada gambar-gambar berikut.

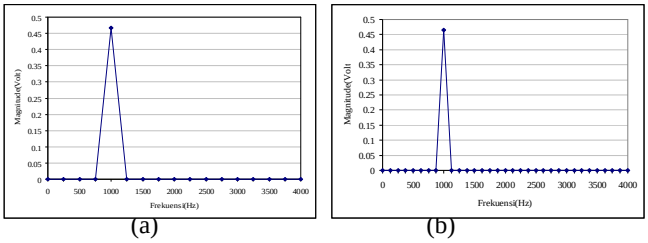


(a) (b)

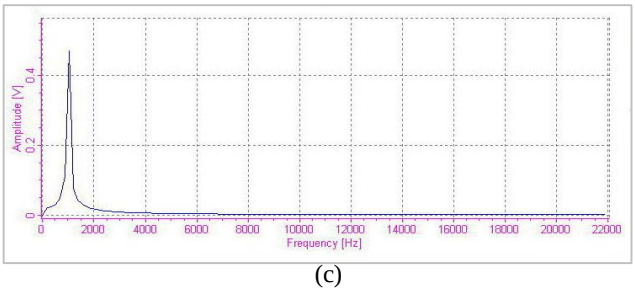


(c)

Gambar 4.2 Tanggapan frekuensi ainyal sinus 500Hz
(a) implementasi 16-titik (b) implementasi 32-titik
(c) FFT Properties

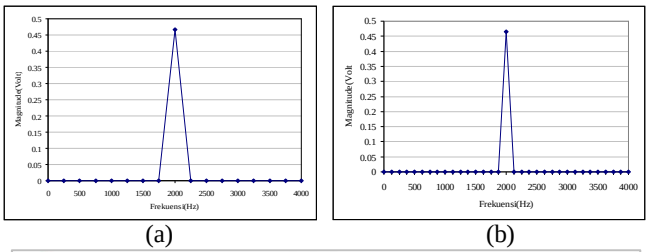


(a) (b)

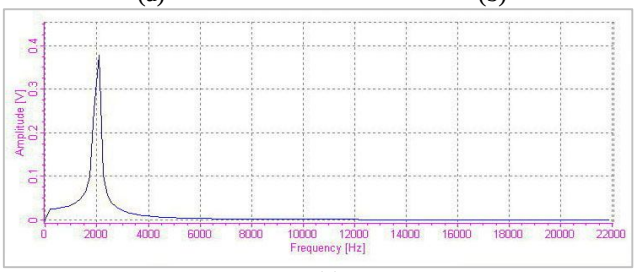


(c)

Gambar 4.3 Tanggapan frekuensi ainyal sinus 1000Hz
(a) implementasi 16-titik (b) implementasi 32-titik
(c) FFT Properties

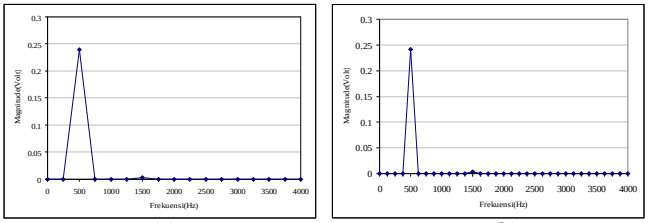


(a) (b)

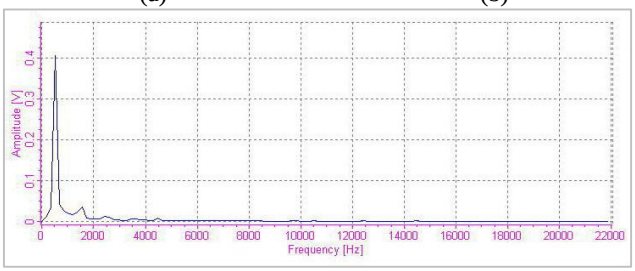


(c)

Gambar 4.4 Tanggapan frekuensi ainyal sinus 1000Hz
(a) implementasi 16-titik (b) implementasi 32-titik
(c) FFT Properties

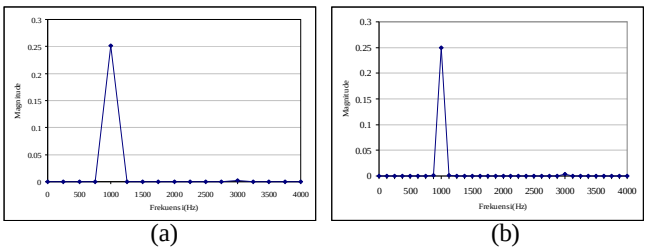


(a) (b)

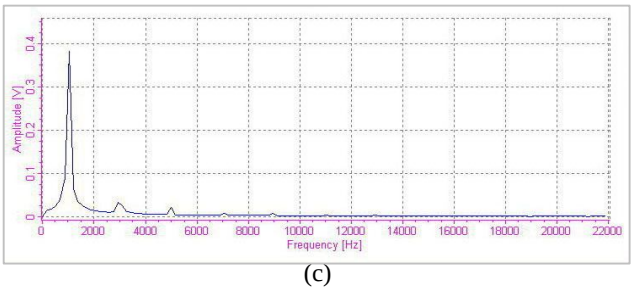


(c)

Gambar 4.5 Tanggapan frekuensi ainyal segitiga 500Hz
(a) implementasi 16-titik (b) implementasi 32-titik
(c) FFT Properties

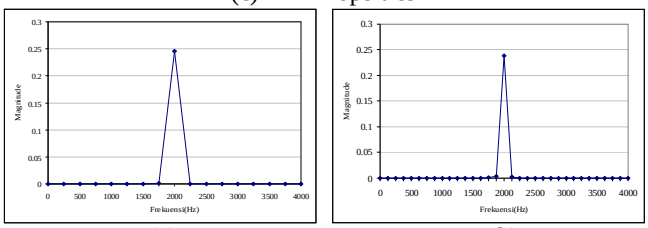


(a) (b)

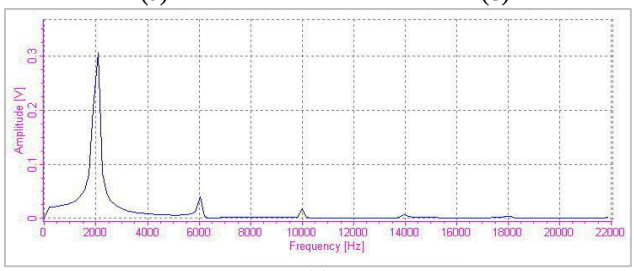


(c)

Gambar 4.6 Tanggapan frekuensi ainyal segitiga 1000Hz
(a) implementasi 16-titik (b) implementasi 32-titik
(c) FFT Properties

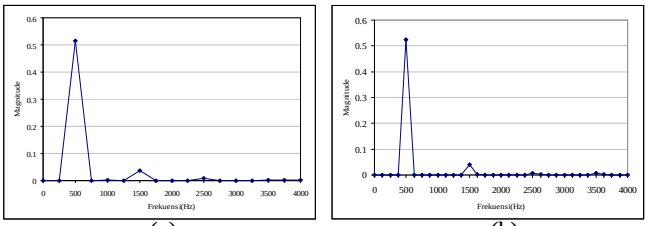


(a) (b)

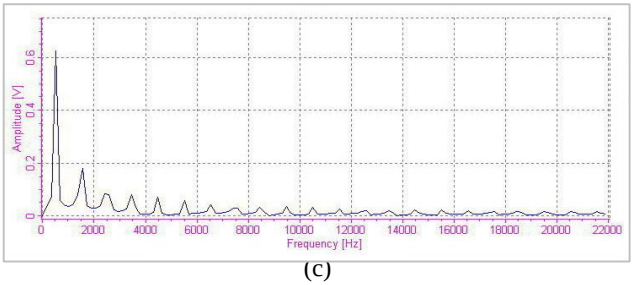


(c)

Gambar 4.7 Tanggapan frekuensi ainyal segitiga 2000Hz
(a) implementasi 16-titik (b) implementasi 32-titik
(c) FFT Properties

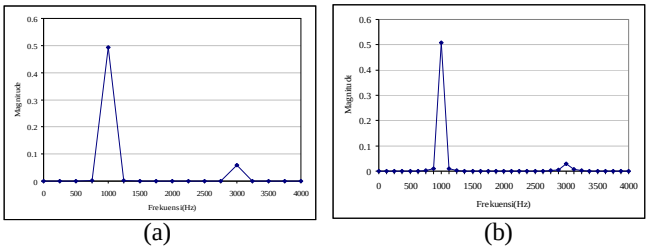


(a) (b)

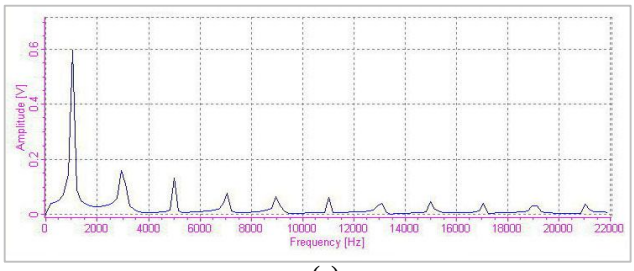


(c)

Gambar 4.8 Tanggapan frekuensi ainyal kotak 500Hz
(a) implementasi 16-titik (b) implementasi 32-titik
(c) FFT Properties

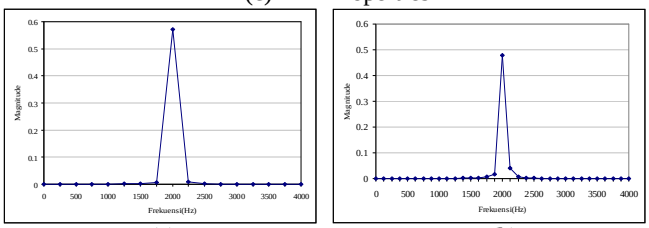


(a) (b)

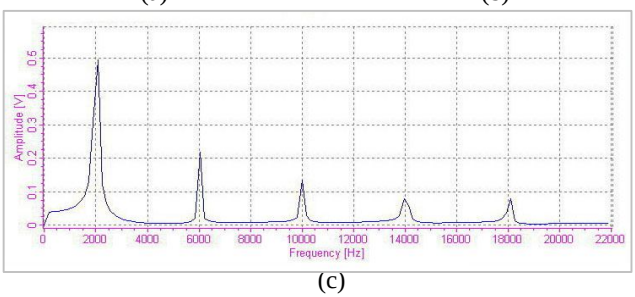


(c)

Gambar 4.9 Tanggapan frekuensi ainyal kotak 1000Hz
(a) implementasi 16-titik (b) implementasi 32-titik
(c) FFT Properties



(a) (b)



(c)

Gambar 4.10 Tanggapan frekuensi ainyal kotak 2000Hz
(a) implementasi 16-titik (b) implementasi 32-titik
(c) FFT Properties

Pada pengujian ditunjukkan bahwa sinyal sinus hanya mempunyai frekuensi fundamental yaitu frekuensi sinyal itu sendiri. Pengujian pada sinyal kotak tampak bahwa nilai harmonisanya lebih jelas dibandingkan pengujian pada sinyal segitiga. Hal tersebut juga terlihat pada hasil simulasi dengan FFT Properties.

Secara teori sinyal segitiga dan sinyal kotak mempunyai komponen-komponen frekuensi (fundamental dan harmonisa) kelipatan tiga, lima, tujuh dan seterusnya dari frekuensi fundamentalnya.

Jumlah cuplikan (2*N*) berpengaruh pada lebar pita (*bandwidth*) nilai-nilai frekuensi pada grafik. Tampak bahwa pada grafik magnitude dengan jumlah cuplikan *N*=8 lebar *bandwidth* yang ditunjukkan adalah 1000Hz lebih besar dibandingkan dengan grafik magnitude dengan jumlah cuplikan yang lebih besar, seperti pada *N*=16 lebar *bandwidth*-nya sama dengan 500Hz dan pada *N*=32 memiliki lebar *bandwidth* 250Hz. Gambar di atas menunjukkan jumlah cuplikan 32 memberikan lebar pita lebih kecil dari pada grafik dengan jumlah cuplikan 16 atau 8, sehingga semakin besar jumlah cuplikan grafik magnitude yang ditunjukkan lebih mendekati nilai teoritis.

Grafik tanggapan frekuensi hasil implementasi terlihat sedikit berbeda dengan grafik tanggapan frekuensi dari simulasi FFT Properties. Hal ini disebabkan oleh perbedaan jumlah cuplikan dan frekuensi cuplik pada program simulasi FFT Properties yaitu sebesar 256 cuplikan dan frekuensi cupliknya 44100Hz sehingga bidang pengamatannya mencapai frekuensi 22KHz.

V. PENUTUP

5.1 Kesimpulan

Setelah dilakukan implementasi algoritma FFT pada DSP TMS320C542 dapat diambil kesimpulan sebagai berikut:

1. Algoritma FFT radiks dua dengan menggunakan metode *Decimation in Time* telah berhasil diimplementasikan pada DSP TMS320C542.
2. Tanggapan frekuensi hasil implementasi telah sesuai dengan teori, yaitu pada sinyal sinus hanya terdapat frekuensi fundamantal saja, sedangkan pada sinyal segitiga dan sinyal kotak terdapat frekuensi fundamental dan frekuensi harmonisa yang muncul pada kelipatan ganjil dari frekuensi fundamental.
3. Pengamatan spektrum frekuensi dibatasi oleh besar frekuensi sampling, di mana pada frekuensi sampling 8KHz pengamatan hanya sampai 4KHz, sehingga harmonisa dengan frekuensi lebih dari 4KHz tidak dapat diamati.
4. Jumlah cuplikan berpengaruh terhadap lebar *bandwidth*, semakin kecil jumlah cuplikan semakin kecil lebar *bandwidth* pada tanggapan frekuensi. Untuk N=8 lebar bandwith adalah 1000Hz, untuk N=16 lebar *bandwidth* adalah 500Hz, dan Untuk N=32 lebar *bandwidth* adalah 250Hz. Semakin besar jumlah cuplikan semakin akurat hasil perhitungan FFT.

5.2 Saran

1. Penggunaan frekuensi *sampling* tinggi akan memperbesar jangkauan pengamatan spektrum frekuensi
2. Perlu dilakukan penelitian lanjutan untuk pengembangan algoritma FFT pada aplikasi-aplikasi dibidang pengolahan suara.

DAFTAR PUSTAKA

[1] Couch II, Leon.W., *Digital and Analog Communication Systems*, Prentice-Hall International, Inc., A Simon & Schuster Company, Upper Saddle River, New Jersey, 1997

[2] Ifeachor, E.C., and Jervis, B.W., *Digital Signal Processing: A Practical Approach*, Addison-Wesley Publishing Company Inc., Wokingham, England, 1993.

[3] Kuc, R., *Introduction to Digital Signal Processing*, McGraw-Hill Book Company, Singapore, 1998

[4] Oppenheim, A.V., and Willsky, A.S., *Sinyal dan Sistem: Jilid 1, Edisi Kedua*, Prentice Hall Inc., A Simon & Schuster/A Viacom Company, New Jersey, 1997.

[5] Proakis, J.G., and Manolakis, D.G., *Pemrosesan Sinyal Digital: Prinsip, Algoritma, dan Aplikasi*, Terjemahan, Prentice Hall Inc., A Simon & Schuster Company, Upper Saddle River, New Jersey, 1995.

[6] Smith, Winthrop W., and Joanne M. Smith, *Handbook of Real-Time Fast Fourier Transform: Algorithms to Product Testing*, IEEE, Inc., New York, 1995.

[7] Valkenburg, M.E.V., *Analisis Jaringan Listrik: Edisi Ketiga*, Addison-Wesley Publishing Company Inc., Wokingham, England, 1993.

[8] ---, *TLC320AC01C Single-Supply Analog Interface Circuit Data Manual*, Texas Instrument Incorporated, Dallas, Texas, USA, 1996.

[9] ---, *TMS320C542 DSKPlus DSP Starter Kit User Guide*, Texas Instrument Incorporated, Owensville, Missouri, USA, 1996.

[10] ---, *TMS320C54x Assembly Language Tools User's Guide*, Texas Instrument Incorporated, Owensville, Missouri, USA, 1997.

[11] ---, *TMS320C54x DSP Reference Set Volume 1: CPU and Peripherals*, Texas Instrument Incorporated, Owensville, Missouri, USA, 1996.

[12] ---, *TMS320C54x DSP Reference Set Volume 3: Algebraic Instruction Set*, Texas Instrument Incorporated, Owensville, Missouri, USA, 1996.

Nandra Pradipta lahir diJakarta, 19 Desember 1982. Lulus dari SMU 78 Jakarta tahun 2000. Saat ini sedang menempuh pendidikan tinggi di Jurusan Teknik Elektro Undip, konsentrasi elektronika dan telekomunikasi.

Menyetujui dan mengesahkan

Pembimbing I	Pembimbing II
Achmad Hidayatno, ST. MT.	Trias Andromeda, ST. MT.
NIP. 132 137 933	NIP. 132 283 185