ISSN: 0852 - 2863

ITS
Institut
Teknologi
Sepuluh Nopember

eepis JOURNAL

electronic, power, control, telecommunication, information and computer

Agustus 2007 Volume 12 Number 1

ITS
Institut
Teknologi
Sepuluh Nopember

Electronic Engineering
Polytechnic Institute of Surabaya

http://www/eepis-its.edu

Journal of

EEPIS
Electrical Electronic Power Information Systems

Editor Kepala
Endra Pitowarno

Electrical
Mauridhi Hery P (ITS)
Pekik Argo D (ITB)
Feri Y (UI)
M. Ashari (ITS)
Adi Soeprijanto (ITS)
Tumiran (UGM)

Electronic
Dedi Wicaksono (TU Delf)
Rusminto Tjatur W (PENS)
Achmad Arifin (ITS)
Djoko Purwanto MEng (ITS)
Brian Yuliarto (ITB)
Nemuel Paah (Ubaya)

Control
Son Kuswadi (PENS)
Bambang Riyanto (ITB)
Ari Santoso (ITS)
Riyanto (LIPI)
Mulyo Widodo (ITB)
Wahyudi (IIUM)

Telkom
Titon Dutono (PENS)
Gamantyo (ITS)
Gunawan W (UI)
Sultan HS (ITB)
Hari Budiarto (BPPT)
Armien Langi (ITB)

Information
Djoko Lianto (ITS)
M. Isa Irawan (ITS)
K. Ramli (UI)
Herlog N. (PolBan)
Benyamin (UI)
Daded Pramadihanto (PENS)

EEPIS Journal (ISSN 0852-2863) is published two times a year in April and December by Electronic Engineering Polytechnic Institute of Surabaya.
Responsibility for the contents rests upon the authors and nout upon the EEPIS.
EEPIS office: Kampus ITS Keputih Sukolilo Surabaya 60111, Indonesia.
Tel +62-31-5947280; ext. 4131, Fax: 62-31-5946114.
E-mail:epit eepis-its.edu, zar eepis-its.edu

DAFTAR ISI

Bambang Sumantri, Ardik Wijayanto
Politeknik Elektronika Negeri Surabaya
Institut Teknologi Sepuluh Nopember
Kampus ITS Keputih Sukolilo Surabaya 60111, Indonesia
E-mail : bambang@eepis-its.edu, ardik@eepis-its.edu

Sebuah Metodologi Pengajaran Pada Eksperimen Pemrosesan Sinyal Digital Menggunakan Modul TMS320VC5402 Dengan Menginvestigasi Orde Filter Maksimum

Hary Oktavianto[1)], Amang Sudarsono[1)], Bima Sena Bayu Dewantara[1)],
Reni Soelistijorini[1)], Miftahul Huda[1)], Tri Budi Santoso[1)], Titon Dutono[1)], Takashi Ikeda[2)]

[1)]Politeknik Elektronika Negeri Surabaya (PENS), Indonesia
Kampus ITS, Keputih-Sukolilo, Surabaya-60111
[2)]Kurume National College of Technology, Japan
E-mail: hary@eepis-its.edu

ABSTRAK

Pada paper ini disajikan sebuah metode pengajaran pada eksperimen pemrosesan sinyal digital di laboratorium. Metode tersebut memanfaatkan algoritma pengolahan sinyal dengan perhitungan bilangan pecahan (floating-point) dan perhitungan bilangan bulat (fixed-point) yang dibenamkan pada sistem prosesor TMS320VC5402 jenis fixed-point. Untuk keperluan pengujian digunakan algoritma filter digital jenis FIR (Finite Impulse Response). Hasil yang didapat menunjukkan bahwa algoritma filter digital menggunakan perhitungan bilangan pecahan masih dapat mencapai real-time pada orde 16. Sedangkan algoritma filter digital yang menggunakan perhitungan bilangan bulat dapat mencapai real-time sampai orde 102. Kedua percobaan tersebut bekerja pada frekuensi sampling 16 kHz. Hal ini dapat menunjukkan kegunaan dari percobaan yang telah dilakukan untuk digunakan pada eksperimen di laboratorium.

Kata Kunci : *digital signal processing, floating-point, fixed-point, finiteiImpulse response.*

1. PENDAHULUAN

Materi perkuliahan pengolahan sinyal umumnya disampaikan dengan memperkenalkan jenis-jenis sinyal, sinyal kontinyu (sinyal analog) dan sinyal diskrit (sinyal digital) beserta operasinya. Selanjutnya bagaimana melakukan proses konversi sinyal analog menjadi sinyal digital dan merekonstruksi kembali sinyal digital menjadi sinyal analog. Berikutnya menyajikan sinyal pada bidang waktu dan bidang frekuensi. Alih ragam Laplace dan Z juga disampaikan masing-masing untuk menyelesaikan persoalan-persoalan pada waktu kontinyu dan waktu diskrit. Alih ragam Z digunakan untuk mencari fungsi alih dan memodelkan sistem seperti sistem filter digital [2],[3],[4],dan [5]. Materi perkuliahan pengolahan sinyal kadang didampingi dengan praktikum pengolahan sinyal. Praktikum pengolahan sinyal di laboratorium dilakukan dengan tujuan untuk memperkenalkan siswa agar mampu membuat desain sistem secara nyata menggunakan perangkat keras. Sebagai contoh, digunakan modul-modul yang menggunakan sistem prosesor pengolah sinyal (DSP: Digital Signal Processor) yang dilengkapi dengan seperangkat komputer beserta pembangkit sinyal dan osiloskop.

DSP merupakan sebuah mikroprosesor dengan arsitektur khusus untuk melakukan perhitungan matematis secara cepat [1],[6]. Di pasar terdapat dua tipe DSP, yaitu tipe *fixed-point* dan tipe *floating-point*. DSP tipe fixed-point hanya dapat mengerjakan perhitungan menggunakan tipe bilangan bulat (*fixed-point*). Sedangkan DSP tipe *floating-point* dapat mengerjakan perhitungan dengan tipe bilangan bulat maupun tipe bilangan pecahan (*floating-point*) karena didalamnya terdapat unit khusus untuk menangani perhitungan bilangan pecahan. Untuk melakukan pemrograman yang efisien dalam bahasa-C lebih mudah dilakukan pada DSP tipe *floating-point* daripada DSP tipe *fixed-point*.

Mendesain program harus memperhatikan tipe prosesor yang dipakai. Kesalahan dalam teknik mendesain program akan memperlambat kinerja dari prosesor. DSP yang digunakan pada paper ini adalah TMS320VC5402. Prosesor tersebut merupakan prosesor 16-bit tipe *fixed-point* dengan frekuensi kerja maksimum sebesar 100MHz. Sedangkan pemrograman prosesor tersebut menggunakan bahasa-C melalui paket program Code Composer Studio®. Percobaan untuk mengetahui dampak teknik pemrograman yang berbeda, yaitu menggunakan perhitungan floating-point dan *fixed-point*, dilakukan dengan cara membenamkan algoritma filter digital FIR kedalam prosesor TMS320VC5402. Pengujian dilakukan dengan cara menghitung lama waktu yang diperlukan untuk menyelesaikan satu kali proses filter pada satu sampel data.

2. FILTER DIGITAL FIR

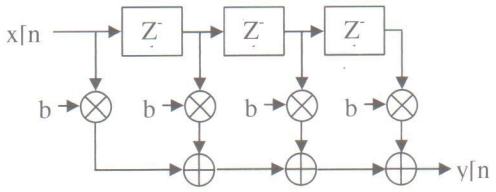
Filter FIR adalah salah satu tipe dari filter digital yang dipakai pada aplikasi DSP. FIR kepanjangan dari *Finite Impulse Response*. Disebut respon impulsnya terbatas karena tidak ada *feedback* didalam filter, jika

dimasukkan sebuah impulse (yaitu sebuah sinyal '1' diikuti dengan banyak sinyal '0'), maka sinyal nol akan keluar setelah sinyal 1 melewati semua *delay line* dengan koefisiennya [2],[3],[4],[5], dan [7].

Keuntungan filter FIR antara lain adalah stabil dan memiliki fase yang linier. Sedangkan kerugiannya adalah filter FIR terkadang membutuhkan lebih banyak memori dan/atau perhitungan untuk mencapai karakteristik respon filter yang diinginkan.

Persamaan filter FIR orde N ditunjukkan oleh persamaan 1 sedangkan *flowgraph* filter FIR orde 3 ditunjukkan oleh Gambar 1.

$$y[n] = \sum_{m=0}^{N-1} b_m x[n-m] \qquad (1)$$



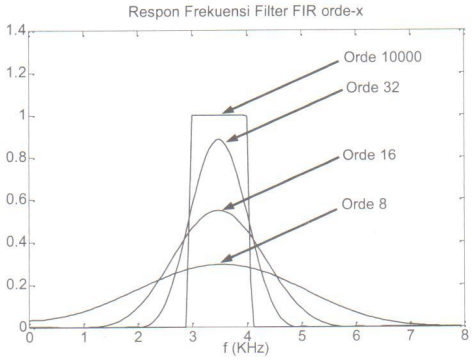
Gambar 1: Flowgraph filter FIR orde 3

Apabila Gambar 1 diterjemahkan kedalam program C maka didapatkan listing program berikut dalam bentuk *subroutine*.

```
int fir(int x)
{
  y = 0;

  // geser delay line
  for(i=orde+1;i>0;i--) {z[i] = z[i-
1];}
  z[0] = x;

  // hitung output
  for(i=0;i<orde+1;i++)      {y      +=
b[i]*z[i];}

  return( (int)y );
}
```

Code Composer Studio® adalah merek dagang dari Texas Instruments.

Semakin besar orde filter semakin mendekati ideal respon frekuensi yang dihasilkan. Pada Gambar 2 ditunjukkan respon frekuensi pada orde yang berbeda. Gambar 2 tersebut dibuat menggunakan MATLAB® untuk kasus filter FIR jenis band-pass dengan frekuensi cut-off 3kHz – 4kHz.



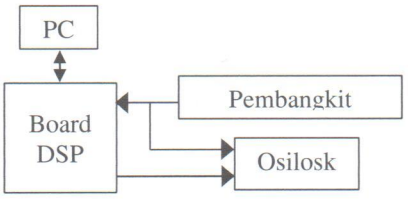
Gambar 2: Contoh, respon frekuensi filter FIR jenis band-pass pada berbagai orde

3. PERSIAPAN PERCOBAAN

Percobaan dilakukan dua bagian yaitu filter FIR yang menggunakan *floating-point* dan filter FIR yang menggunakan *fixed-point*. *Compiler* akan menganggap perhitungan tersebut *floating-point* apabila didalam program digunakan tipe variabel bilangan pecahan seperti *float* atau *double*. Sedangkan penggunaan tipe variabel bilangan bulat seperti *int* dan *long* menyebabkan *compiler* menganggap perhitungan sebagai fixed-point. *Compiler* bertugas menerjemahkan bahasa-C ke bahasa assembly untuk prosesor DSP yang digunakan.

Gambar 3: Peralatan yang digunakan

Peralatan yang digunakan ditunjukkan oleh Gambar 3, dan secara diagram dapat dilihat pada Gambar 4. PC (*Personal Computer*) dipakai untuk mendesain program yang akan dibenamkan kedalam DSP. Setelah program dibenamkan dan dijalankan, sinyal input didapat dari sebuah pembangkit sinyal dan sinyal output hasil proses diamati pada osiloskop.



Gambar 4: Diagram peralatan

Pengukuran lama waktu yang diperlukan dilakukan dengan cara men-*toggle* bit XF dimana bit XF ini terhubung dengan pin prosesor DSP. Pin XF merupakan pin output serba guna. Pin XF ini selanjutnya dihubungkan ke osiloskop untuk diukur rentang waktunya.

Untuk melakukan *toggle* digunakan instruksi set bit dan reset bit dalam bahasa assembly. Baris program yang akan diukur waktu proses eksekusinya "dibungkus" dengan instruksi berikut. Tanda titik sebanyak tiga menunjukkan bahwa pada lokasi tersebut terdapat baris instruksi atau angka yang tidak ditampilkan agar program tidak tampak terlalu panjang.

```
asm(" SSBX XF"); // XF=1
...
// subrutin filter FIR
...
asm(" RSBX XF"); // XF=0
```

Selama proses pengujian, DSP diatur agar bekerja pada frekuensi maksimum yaitu 100MHz sedangkan frekuensi sampling dari codec diatur sebesar 16kHz.

Tabel 1: Spesifikasi peralatan

Peralatan	Spesifikasi
PC	Intel Pentium 4 (2.26GHz) RAM 256Mb OS WindowsXP SP2
Pembangkit Sinyal	IWATSU SG-4105 DDS (Direct Digital Synthesis) Fmax 15 MHz, accuracy ±50 ppm. Output max.±10Volt
Osiloskop	KENWOOD DCS-8300 DC – 20MHz (-3dB) at 1 or 2 mV/division
Board DSP	DSK TMS320C5402

4. HASIL PERCOBAAN DAN ANALISIS

Operasi filter FIR akan menjumlahkan hasil perkalian antara sinyal input dengan koefesien filter. Sinyal input didapat dari hasil konversi analog ke digital (dari ADC) yang mempunyai tipe bilangan bulat bertanda (*signed integer*). Koefesien filter didapat dengan bantuan Matlab. Nilai yang didapat biasanya dalam pecahan. Jenis filter yang digunakan pada percobaan ini adalah filter low-pass dengan frekuensi cut-off sebesar 3 kHz.

4.1. Percobaan filter FIR *floating-point*

Nilai koefesien filter yang didapat dari Matlab dapat langsung dimasukkan kedalam program filter menggunakan tipe bilangan *float*. Variabel y tempat menampung hasil perhitungan juga menggunakan tipe bilangan *float*.

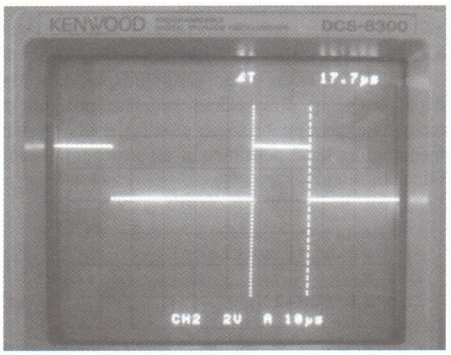
```
...
// definisi variabel
int x;                    // input
int z[orde+1];            //      delay
line
float y;                  // output
float b[orde+1]={              //
koefesien
          -0.002273,
          ...,
          0.002273};
```

Pengukuran lama waktu eksekusi menggunakan bit XF diletakkan pada bagian utama program sebagai berikut.

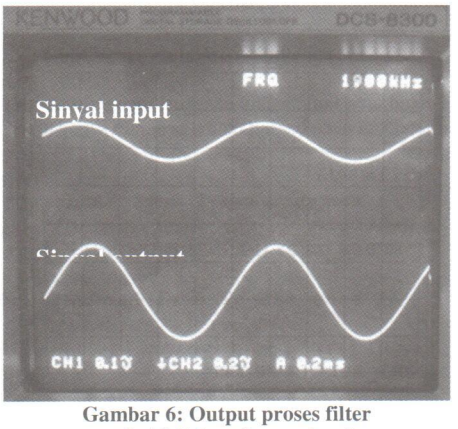
```
while (1)
{ /* tunggu sampel dari handset */
  while
(!MCBSP_RRDY(HANDSET_CODEC)) {};

  /* baca sampel dari handset codec
*/
  x      =      *(volatile      u16*)
DRR1_ADDR(HANDSET_
  CODEC);

   asm(" SSBX XF"); // XF=1

  y = 0;
   // geser delay line
   for(i=orde+1;i>0;i--)    {z[i]    =
z[i-1];}
    z[0] = x;
    // hitung output
   for(i=0;i<orde+1;i++)     {y     +=
b[i]*z[i];}
   x=(int)y;
```

```
  asm(" RSBX XF"); // XF=0

  /* tulis output ke handset codec
*/
  *(volatile
u16*)DXR1_ADDR(HANDSET_CODEC) =
  x;}
```

Gambar 5 menunjukkan hasil pengukuran sinyal XF.

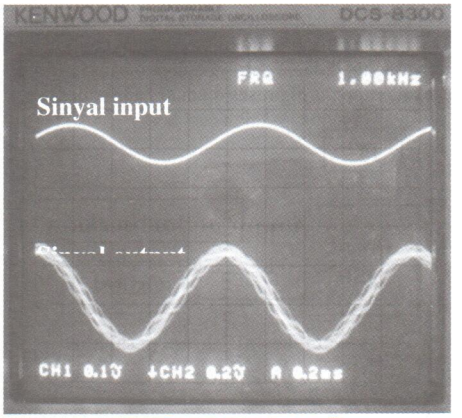


Gambar 5: Pengukuran sinyal XF



Gambar 6: Output proses filter orde 16 didaerah pass-band

Gambar 6 menunjukkan hasil pengukuran sinyal input dan sinyal output pada daerah *pass-band* (daerah lolos) pada orde 16, dimana proses masih *real-time*. Gambar 7 menunjukkan hasil pengukuran sinyal input dan sinyal output pada daerah *pass-band* dimana proses sudah tidak mencapai *real-time*. Lama waktu eksekusi pada orde 4 sampai orde 32 pada filter FIR menggunakan perhitungan *floating-point* ditunjukkan oleh Tabel 1. Sinyal output dikatakan baik apabila hasilnya sama dengan sinyal inputnya seperti pada Gambar 6. Sinyal output dikatakan cacat apabila sudah tidak *real-time* seperti pada Gambar 7.



Gambar 7: Output proses filter yang sudah tidak *real-time*

Berdasar hasil pengukuran waktu eksekusi pada Tabel 2 menunjukkan bahwa mulai orde 17, filter FIR sudah tidak dapat melakukan proses secara *real-time*. Analisanya dapat dijelaskan sebagai berikut.

Frekuensi sampling yang digunakan adalah 16kHz sehingga periode sampling adalah sebesar 62.5 µdetik. Hal ini berarti bahwa jarak tiap sampel adalah 62.5 µdetik.

Tabel 2: Waktu eksekusi menggunakan perhitungan floating-point

Orde filter	Waktu (µs)	Sinyal output
4	17.7	Baik
8	32.0	Baik
16	61.6	Baik
17	65.2	Cacat
18	67.2	Cacat
20	74.6	Cacat
22	80.4	Cacat
24	87.0	Cacat
26	93.2	Cacat
28	99.4	Cacat
30	106.4	Cacat
32	114.5	Cacat

Artinya adalah proses perhitungan haruslah selesai sebelum sampel berikutnya muncul, dengan kata lain waktu eksekusi tidak boleh melebihi 62.5 µdetik. Apabila proses perhitungan lebih lama dari jarak tiap sampel maka sistem menjadi tidak *real-time* karena ada beberapa sampel yang terbuang (tidak diproses) sehingga proses rekonstruksi sinyal menjadi salah. Pertanyaan yang muncul adalah mengapa hasil

pengujian perhitungan menggunakan *floating-point* membutuhkan waktu yang cukup lama?

```
{
      y += b[i]*z[i];
      0000:0236 7211       MVDM   1058h,11h
      0000:0238 F495       NOP
      0000:0239 10E1       LD     *AR1(4186),A
      0000:023B F074  ⇨    CALL   F$$ITOF
      0000:023D 4E00       DST    A,.data
      0000:023E 6FF8       LD     *(1058h),1,A
      0000:0241 8811       STLM   A,11h
      0000:0242 F495       NOP
      0000:0243 F495       NOP
      0000:0244 56E1       DLD    *AR1(4222),A
      0000:0246 F074  ⇨    CALL   F$$MUL
      0000:0248 4E00       DST    A,.data
      0000:0249 56F8       DLD    *(1056h),A
      0000:024B F074  ⇨    CALL   F$$ADD
      0000:024D 4EF8       DST    A,*(1056h)
}
```

Hasil kompilasi dalam bahasa assembly pada inti program filter FIR dapat dilihat pada cuplikan program diatas. Dapat dilihat bahwa terdapat pemanggilan rutin-rutin `CALL F$$ITOF`, `CALL F$$MUL` dan `CALL F$$ADD` (diberi tanda panah kanan). Rutin ini terdapat didalam library *rts.lib* yang digunakan dalam proses kompilasi. Header dari rutin `F$$ITOF` adalah sebagai berikut yang bertugas untuk melakukan konversi bilangan bulat bertanda (*signed integer*) 16-bit menjadi bilangan pecahan (*floating-point*) 32-bit. Rutin `F$$MUL` digunakan untuk melakukan perkalian bilangan pecahan dan rutin `F$$ADD` digunakan untuk melakukan penjumlahan bilangan pecahan.

```
;* F$$ITOF  convert a 16 bit signed integer to floating point
;* FITOF v5.79
;* Copyright (c) 1995 2001 Texas Instruments Incorporated
;*****************************************************************
;*                                                               *
;* revision:  original                                           *
;*                                                               *
;*****************************************************************
;*                                                               *
;*      F$$ITOF                                                  *
;*                                                               *
;*      this routine converts a 16-bit signed                    *
;*      integer to a 32-bit floating point                       *
;*      upon entry the integer is in accumulator                 *
;*      A.  when the conversion is complete                      *
;*      the float value will be in                               *
;*      accumulator A                                            *
;*                                                               *
;*      inputs:  A (floating point value) on stack               *
;*                                                               *
;*      implementation:  The absolute value of the               *
;*          integer is normalized in the exponent.               *
;*          The exponent is determined by decrementing           *
;*          from the assumed maximum value.  The                 *
;*          sign, exponent, and normalized mantissa              *
;*          (imp'ied one's LSB removed) are packed               *
;*          back into accumulator A                              *
;*                                                               *
;*      results:  returned in the accumulator                    *
;*                                                               *
```

Dapat disimpulkan ternyata bila perhitungan *floating-point* dipaksakan pada DSP tipe *fixed-point* maka akan dilakukan konversi tipe float ke tipe integer terlebih dahulu karena pada prosesor tersebut memang tidak tersedia unit yang bertugas khusus untuk menangani operasi bilangan pecahan. Hal inilah yang menyebabkan adanya tambahan waktu eksekusi.

4.2. Percobaan filter FIR *fixed-point*

Agar diperoleh nilai koefesien filter bertipe bilangan bulat maka nilai koefesien filter yang didapat dikalikan dengan 2^{15} atau 32768. Hasil perkalian yang didapat dibulatkan ke nilai bawah. Didalam program, tipe variabel yang digunakan juga bertipe bilangan bulat sehingga bagian deklarasi variabel dibuat sebagai berikut.

```
  ...
  // definisi variabel
  int x;                    // input
  int z[orde+1];            //      delay
line
  long int y;                     //
  output
  int b[orde+1]={           // koefesien
            -74,
            ...,
            74};
```

Variabel y sebagai penampung hasil perhitungan menggunakan tipe *long int* (32-bit) agar dapat menampung hasil perkalian antara koefesien dengan *delay line* yang bertipe *int* (16-bit). Setelah perhitungan selasai, variabel y dibagi kembali dengan 32768 agar didapatkan nilai bilangan selebar 16-bit. Pembagian tersebut juga agar didapatkan nilai yang benar karena sebelumnya nilai koefesien telah dilakukan skala dengan 32768. Program utama tidak berubah seperti pada pengujian *floating-point* sebelumnya.

```
  while (1)
  {
    /* tunggu sampel dari handset */
    while
  (!MCBSP_RRDY(HANDSET_CODEC)) {};

    /* baca sampel dari handset codec
  */
    x      =      *(volatile      ul6*)
  DRR1_ADDR(HANDSET_
    CODEC);

     asm(" SSBX XF"); // XF=1

    y = 0;
     // geser delay line
     for(i=orde+1;i>0;i--)    {z[i]    =
  z[i-1];}
     z[0] = x;
     // hitung output
     for(i=0;i<orde+1;i++)     {y     +=
  b[i]*z[i];}
     x=y>>15;
```

```
    asm(" RSBX XF"); // XF=0

    /* tulis output ke handset codec
*/
    *(volatile
u16*)DXR1_ADDR(HANDSET_CODEC) =
    x;
}
```

Lama waktu eksekusi pada orde 4 sampai orde 120 pada filter FIR menggunakan perhitungan fixed-point ditunjukkan oleh Tabel 3. Sinyal output dikatakan baik apabila hasilnya sama dengan sinyal masukan seperti pada gambar 6. Sinyal output dikatakan cacat apabila sudah tidak *real-time* seperti pada gambar 7.

Berdasar dari hasil pengukuran waktu eksekusi pada Tabel 3 menunjukkan bahwa mulai orde 102, filter FIR sudah tidak dapat melakukan proses secara *real-time*. Analisanya sama dengan pada pengujian dengan perhitungan *floating-point* sebelumnya. Lalu mengapa hasil pengujian perhitungan menggunakan *fixed-point* dapat bekerja sampai orde 102?

Tabel 3: Waktu eksekusi menggunakan perhitungan fixed-point

Orde filter	Waktu (µs)	Sinyal output
4	3.05	Baik
8	5.40	Baik
16	10.2	Baik
32	19.6	Baik
64	38.8	Baik
80	47.4	Baik
100	59.4	Baik
102	61.4	Baik
103	63.0	Cacat
110	65.4	Cacat
120	71.2	Cacat

Apabila hasil compiler pada bagian inti proses perhitungan filter FIR dalam bahasa assembly dilihat sebagai berikut.

```
{
    y += b[i]*z[i];
    0000:0235 7211      MVDM   0eceh,11h
    0000:0237 F495      NOP
    0000:0238 30E1      LD     *AR1(3792),T
    0000:023A 20E1      MPY    *AR1(3992),A
    0000:023C 50F8      DADD   *(0ecch),A,A
    0000:023E 4EF8      DST    A,*(0ecch)
}
```

Dengan menggunakan tipe bilangan bulat pada semua variabel yang digunakan ternyata *compiler* tidak menambahkan pemanggilan subrutin untuk melakukan perhitungan karena instruksi assembly yang ada sudah mampu untuk melakukannya.

5. KESIMPULAN

Hasil percobaan untuk mencari orde maksimum pada kasus filter FIR pada DSP 16-bit *fixed-point* TMS320VC5402 dengan frekuensi sampling 16kHz dalam grafik ditunjukkan oleh Gambar 8.

Karena pada DSP tipe fixed-point tidak terdapat unit yang berfungsi untuk melakukan perhitungan bilangan pecahan maka orde filter pada perhitungan fixed-point lebih besar dibandingkan dengan orde filter pada perhitungan floating-point. Algoritma yang ditanamkan pada prosesor DSP tipe fixed-point dapat bekerja cepat apabila didalam program hanya menggunakan variabel dengan tipe bilangan bulat. Penggunaan perhitungan bilangan pecahan akan menyebabkan tambahan subrutin untuk konversi bilangan yang menyebabkan waktu eksekusi bertambah.

Gambar 8: Orde maksimum filter FIR

Beberapa aplikasi membutuhkan perhitungan *floating-point* atau *fixed-point* atau keduanya dengan mempertimbangkan tingkat akurasi perhitungan dan kecepatan proses.

6. UCAPAN TERIMA KASIH

Paper ini adalah hasil yang dicapai pada penelitian short-term JICA expert bulan Maret hingga Agustus 2006. Terima kasih kepada JICA dan Profesor Takashi Ikeda atas bimbingannya. Terima kasih juga kepada seluruh counterpart yang namanya dicantumkan pada paper ini. Terima kasih kepada Pak Heru dan Pak Madiono atas bantuan fasilitasnya. Juga terima kasih kepada Dr. Endra Pitowarno atas bantuan penulisan abstrak dan tata tulis yang lebih baik.

PUSTAKA ACUAN

[1] Berkeley, "Choosing a DSP Processor", California: Berkeley Design Technology, Inc., http://www.BDTI.com, 2000.

[2] Langi, A. Z. R., "Dasar-Dasar Pemrosesan Sinyal Digital", Diktat Kuliah EL 302 Pemrosesan Sinyal Digital (Ed. III). Indonesia: Institut Teknologi Bandung, 2001.

[3] Ludeman, L.C., "Fundamental of Digital Signal Processing", New York: Harper & Row Publishers, 1986

[4] Oppenheim, A.S., Willsky, A.S., Nawab, S. H., "Sinyal & Sistem", Jilid 2 (Ed.2), Jakarta: Penerbit Erlangga, 1997.

[5] Smith, S. W., "The Scientist and Engineer's Guide to Digital Signal Processing Second Edition", San Diego, California: California Technical Publishing, 1999.

[6] Texas Instruments, "SPRS079D : TMS320 VC5402 Fixed-Point Digital Signal Processor datasheet", USA: Texas Instruments, 2000.

[7] Wikipedia,"Finite Impulse Response", (http://en.wikipedia.org/wiki/Finite_impulse_response, diakses tanggal 16 Nopember 2006).

BIOGRAFI

Hary Oktavianto, dilahirkan di Surabaya 1 Oktober 1976. Menyelesaikan Sarjana di Jurusan Teknik Elektro-Elektronika ITS pada tahun 2001. Sejak 2002 menjadi staf pengajar di PENS Jurusan Elektronika dan riset yang ditekuni adalah Embedded System for Digital Signal Processing.

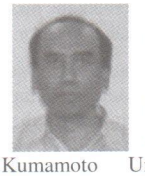
Amang Sudarsono, dilahirkan di Surabaya 20 September 1974. Menyelesaikan Sarjana di Jurusan Teknik Elektro-Telekomunikasi ITS pada tahun 2001. Sejak 2002 menjadi staf pengajar di PENS Jurusan Telekomunikasi dan riset yang ditekuni adalah Jaringan Komputer dan Multimedia.

Bima Sena Bayu Dewantara, dilahirkan di Malang 15 Oktober 1976. Menyelesaikan Sarjana di Jurusan Teknologi Informasi PENS pada tahun 2004. Sejak 1998 menjadi staf pengajar di PENS Jurusan Elektronika dan riset yang ditekuni adalah Jaringan Syaraf Tiruan dan Image Processing.

Reni Sulistijorini, dilahirkan di Surabaya 28 April 1971. Menyelesaikan Sarjana dan Magister di Nagaoka University of Technology, Jepang dan Teknik Elektro-Telekomunikasi Multimedia ITS pada tahun 1998 dan 2005. Sejak 1998 menjadi staf pengajar di PENS Jurusan Telekomunikasi dan riset yang ditekuni adalah Speech Processing.

Miftahul Huda, dilahirkan di Lamongan 12 Oktober 1963. Menyelesaikan Sarjana dan Magister di Jurusan Teknik Elektro-Telekomunikasi Multimedia ITS pada tahun 1990 dan 2002. Sejak 1990 menjadi staf pengajar di PENS Jurusan Telekomunikasi dan riset yang ditekuni adalah Speech Processing

Tri Budi Santoso, dilahirkan di Tulungagung 05 Januari 1970. Meyelesaikan Sarjana dan Magister di Jurusan Teknik Fisika dan Jurusan Teknik Elektro-Telekomunikasi ITS pada tahun 1994 dan 1999. Sejak 1995 menjadi staf di PENS Jurusan Telekomunikasi dan riset yang ditekuni adalah Speech Processing.



Titon Dutono, dilahirkan di Surabaya 30 Nopember 1960. Menyelesaikan Sarjana di Jurusan Teknik Elektro ITS pada tahun 1985. Tahun 1996 menyelesaikan Magister di Kumamoto University, Jepang. Tahun 1999 menyelesaikan program Doktor ditempat yang sama dengan spesialisasi Speech Processing.

Takashi Ikeda Profesor di Kurume National College of Technology, Jepang. Banyak mengerjakan bidang pengolahan sinyal wicara waktu nyata. Beberapa kali datang di PENS sebagai expert dengan dukungan JICA.