[A. pen IC] FFT C language implementation

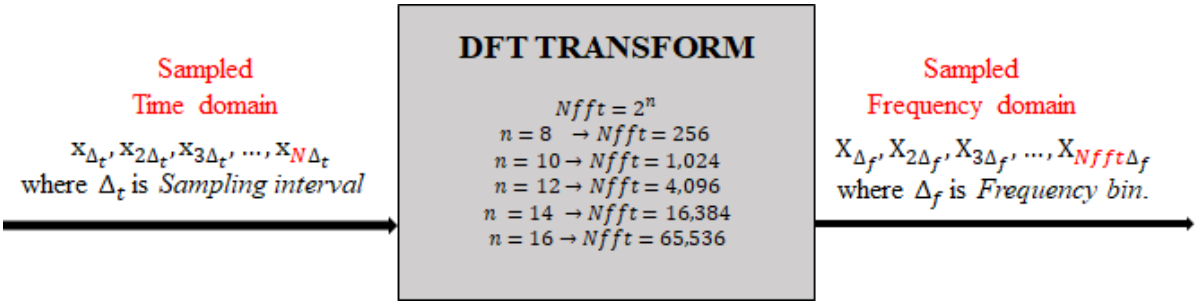
Abstract:

This article aims to present the Fast Fourier transform by C language.

1. -A novel Radix-2 butterfly + decimation in frequency will be used.
2. -GNU C language is used to implement those modules: Twiddle factors, Radix-2, Ordering module.
3. -The program is designed to read time data from txt file, and outputs frequency data in also txt file.

FFT algorithm and C language Implementation:

- View of FFT:



- Complex number and twiddle factors:

$$W_N = e^{-j\frac{2\pi}{N}} = \cos\left(\frac{2\pi}{N}\right) - j\sin\left(\frac{2\pi}{N}\right) \rightarrow W_N^{kt} = e^{-j\frac{2\pi}{N}kt}$$
$$= \cos\left(\frac{2\pi}{N}kt\right) - j\sin\left(\frac{2\pi}{N}kt\right)$$

```
10  #include <stdio.h>
11  #include <stdlib.h>
12  #include <math.h>
13  ////////////////////////////
14  #define PI 3.14159265359
15  #define MAXPOW 24
```

```
16  struct complex
17  {
18      double r;
19      double i;
20  };
21  int pow_2[MAXPOW];
22  ////////////////////////////////////////////////////////////
23  void twiddle(struct complex *W, int N, double stuff)
24  {
25      W->r=cos(stuff*2.0*PI/(double)N);
26      W->i=-sin(stuff*2.0*PI/(double)N);
27  }
```

- General equations as Radix-2, Decimation on frequency algorithm:

$$X[k_1 + 2k_2] = \sum_{n_2=0}^{N/2-1} [(x[n_2] + (-1)^{k_1} x[N/2 + n_2]) W_N^{k_1 n_2}] W_{N/2}^{k_2 n_2}$$

```c
//////////////////////////////////////////////////////////////
/** RADIX-2 FFT ALGORITHM */
void radix2(struct complex *data, int N)
{
    int    n2, k1, N1, N2;
    struct complex W, bfly[2];
    N1  =   2;
    N2  =   N/2;
    /** Do 2 Point DFT */
    for (n2=0; n2<N2; n2++)
    {
        /** Don't hurt the butterfly */
        twiddle(&W, N, (double)n2);
        bfly[0].r = (data[n2].r + data[N2 + n2].r);
        bfly[0].i = (data[n2].i + data[N2 + n2].i);
        bfly[1].r = (data[n2].r - data[N2 + n2].r) * W.r
                  - ((data[n2].i - data[N2 + n2].i) * W.i);
        bfly[1].i = (data[n2].i - data[N2 + n2].i) * W.r
                  + ((data[n2].r - data[N2 + n2].r) * W.i);
        /** In-place results */
        for (k1=0; k1<N1; k1++)
        {
            data[n2 + N2*k1].r = bfly[k1].r;
            data[n2 + N2*k1].i = bfly[k1].i;
        }
    }
    /** Don't recurse if we're down to one butterfly */
    if (N2!=1)
    {
        for (k1=0; k1<N1; k1++)
            radix2(&data[N2*k1], N2);
    }
}
```

- Purpose of ordering:

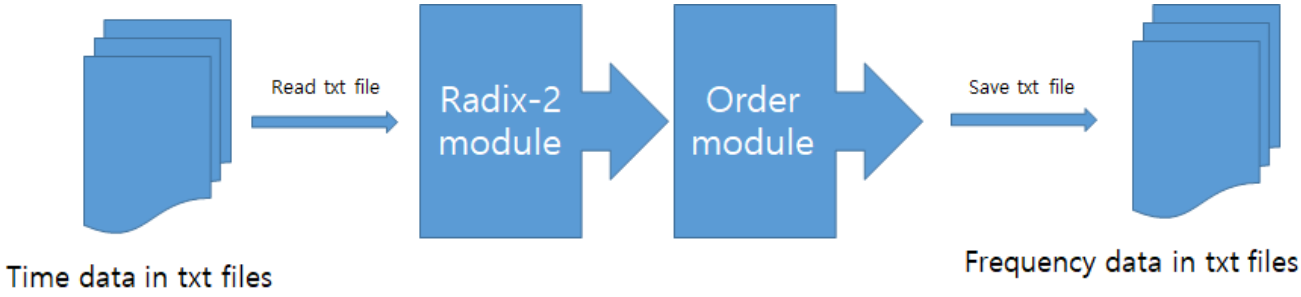
Input order		Output order	
Num.	binary	binary	Num.
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

```c
//////////////////////////////////////////////////////////////
void bit_reverse_reorder(struct complex *W, int N)
{
    int     bits, i, j, k;
    double  tempr, tempi;

    for (i=0; i<MAXPOW; i++)
    {
        if (pow_2[i]==N)
            bits=i;
    }

    for (i=0; i<N; i++)
    {
        j=0;
        for (k=0; k<bits; k++)
        {
            if (i&pow_2[k])
            {
                j+=pow_2[bits-k-1];
            }
        }
        if (j>i)  /** Only make "up" swaps */
        {
            tempr   =   W[i].r;
            tempi   =   W[i].i;
            W[i].r  =   W[j].r;
            W[i].i  =   W[j].i;
            W[j].r  =   tempr;
            W[j].i  =   tempi;
        }
    }
```

- Main module:



Attachment:

- ppt file for full explanation
- C language source code
- Test input data

- Output results


```
[mducng@lsi1: /user/mducng/fftC]pwd
/user/mducng/fftC
[mducng@lsi1: /user/mducng/fftC]ls -l
total 24
-rw-r--r-- 1 mducng design 3618 Nov  9 14:43 myFFTv4.c
-rw-r--r-- 1 mducng design 9723 Nov  9 14:43 test1024f.txt
-rw-r--r-- 1 mducng design  300 Nov  9 14:43 test32f.txt
-rw-r--r-- 1 mducng design  603 Nov  9 14:43 test64f.txt
[mducng@lsi1: /user/mducng/fftC]gcc -lm myFFTv4.c -o fft4
[mducng@lsi1: /user/mducng/fftC]ls -l
total 36
-rwxr-xr-x 1 mducng design 10170 Nov  9 14:44 fft4
-rw-r--r-- 1 mducng design  3618 Nov  9 14:43 myFFTv4.c
-rw-r--r-- 1 mducng design  9723 Nov  9 14:43 test1024f.txt
-rw-r--r-- 1 mducng design   300 Nov  9 14:43 test32f.txt
-rw-r--r-- 1 mducng design   603 Nov  9 14:43 test64f.txt
[mducng@lsi1: /user/mducng/fftC]fft4 test32f.txt 32
[mducng@lsi1: /user/mducng/fftC]ls -l
total 44
-rwxr-xr-x 1 mducng design 10170 Nov  9 14:44 fft4
-rw-r--r-- 1 mducng design  3618 Nov  9 14:43 myFFTv4.c
-rw-r--r-- 1 mducng design   312 Nov  9 14:44 outFFT_image.txt
-rw-r--r-- 1 mducng design   328 Nov  9 14:44 outFFT_real.txt
-rw-r--r-- 1 mducng design  9723 Nov  9 14:43 test1024f.txt
-rw-r--r-- 1 mducng design   300 Nov  9 14:43 test32f.txt
-rw-r--r-- 1 mducng design   603 Nov  9 14:43 test64f.txt
```

BRs,

mducng