uProcessor Term Project: FFT & MVM optimization

10조: 이승훈. 박진호
건국대학교
전기전자공학부

FFT Optimization result

```
<4096-point Fourier Transform>

Measured Accuracy: NSR(dB) = -123.045

-----Benchmarking Start-----
Case 0: DFT Reference
Nr,         Max,         Min,     Average,    Fltr Avg,  Fltr_Avg(ms)
 5,  3115581451,  3115570290,  3115574662,  3115573857,     9346.721
Case 1: FFT Optimization
Nr,         Max,         Min,     Average,    Fltr Avg,  Fltr_Avg(ms)
 5,      391250,      387395,      388634,      388176,        1.165
----Benchmarking Complete----

Optimized FFT is x7963.15 faster than Reference
```

- 2가지 방법을 통해서 개선했습니다.
  - digit reverse algorithm
  - Radix 4 Decimation in Frequency FFT

Digit reverse algorithm

```
unsigned char reverse_table_2bits[64] ={
        0x0,0x10,0x20,0x30,0x4,0x14,0x24,0x34,
        0x8,0x18,0x28,0x38,0xc,0x1c,0x2c,0x3c,
        0x1,0x11,0x21,0x31,0x5,0x15,0x25,0x35,
        0x9,0x19,0x29,0x39,0xd,0x1d,0x2d,0x3d,
        0x2,0x12,0x22,0x32,0x6,0x16,0x26,0x36,
        0xa,0x1a,0x2a,0x3a,0xe,0x1e,0x2e,0x3e,
        0x3,0x13,0x23,0x33,0x7,0x17,0x27,0x37,
        0xb,0x1b,0x2b,0x3b,0xf,0x1f,0x2f,0x3f,
};

for(n=0; n < N;n++){
   out[n] = input[reverse_table_2bits[n >> 6] | reverse_table_2bits[n & 0b111111] << 6];
}
```

- 기존의 FFT알고리즘을 분석해본결과 bit reverse에서 처리시간이 큰 것을 확인 할 수 있었습니다.
- 사용되는 연산시간을 줄이기 위해 미리 lookup table을 만들어서 bit reverse를 했습니다.
- reverse_table_2bits은 64개의 1byte element를 가지는 배열입니다. 그대로 사용하면 배열공간이 너무 커지기 때문에 반인 size를 64로 했습니다.

Radix 4 Decimation in Frequency FFT

N	Radix-2	Radix-4
16	24	20
32	88	
64	264	208
128	72	
256	1800	1392
512	4360	
1024	10248	7856

- **Multiplication연산량을 비교한 결과입니다.**
  - **radix 2 Decimation in Frequency FFT에 비해서 stage 수가 줄어들어서 성능향상을 가져올수 있습니다.**

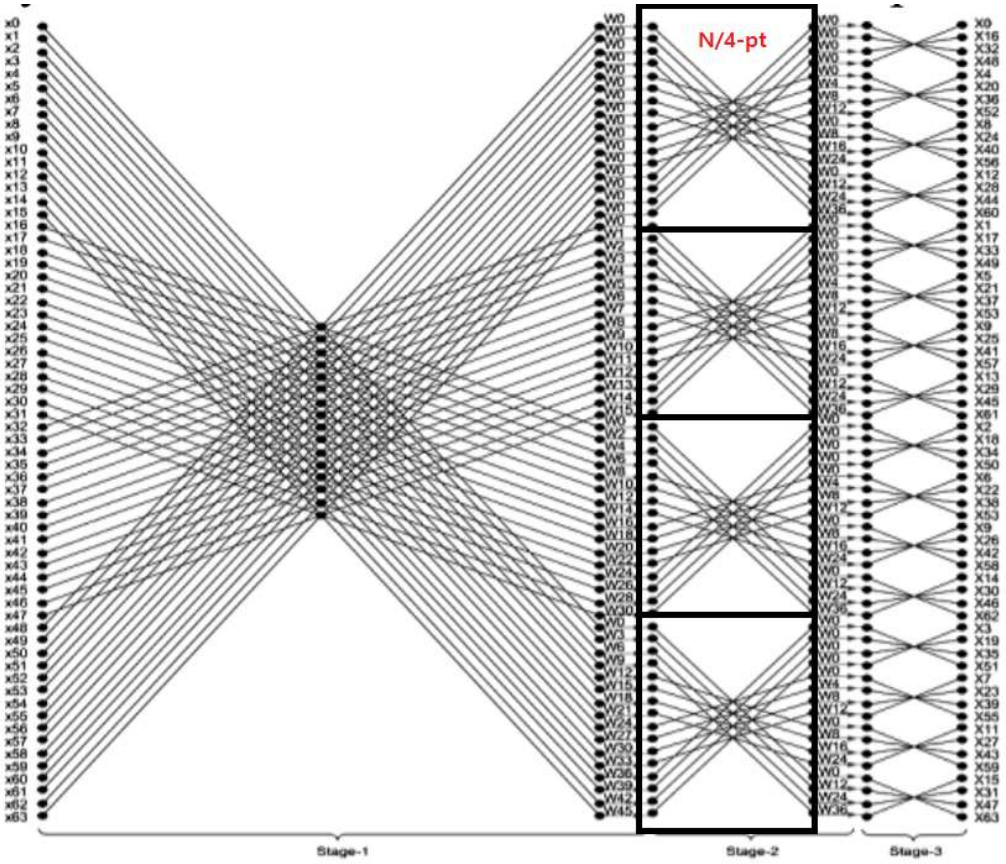
Radix 4 Decimation in Frequency FFT(cont)

$$
\begin{aligned}
X(k) &= \sum_{n=0}^{N-1} x(n) W_N^{kn} \\
&= \sum_{n=0}^{N/4-1} x(n) W_N^{kn} + \sum_{n=N/4}^{N/2-1} x(n) W_N^{kn} + \sum_{n=N/2}^{3N/4-1} x(n) W_N^{kn} + \sum_{n=3N/4}^{N-1} x(n) W_N^{kn} \\
&= \sum_{n=0}^{N/4-1} x(n) W_N^{kn} + W_N^{Nk/4} \sum_{n=0}^{N/4-1} x\left(n+\frac{N}{4}\right) W_N^{kn} + \\
&\quad W_N^{Nk/2} \sum_{n=0}^{N/4-1} x\left(n+\frac{N}{2}\right) W_N^{kn} + W_N^{3Nk/4} \sum_{n=0}^{N/4-1} x\left(n+\frac{3N}{4}\right) W_N^{kn}
\end{aligned}
$$

$$
\begin{aligned}
X(4k) &= \sum_{n=0}^{N/4-1} \left[x(n) + x\left(n+\frac{N}{4}\right) + x\left(n+\frac{N}{2}\right) + x\left(n+\frac{3N}{4}\right) \right] W_N^{0} W_{N/4}^{kn} \\
X(4k+1) &= \sum_{n=0}^{N/4-1} \left[x(n) - jx\left(n+\frac{N}{4}\right) - x\left(n+\frac{N}{2}\right) + jx\left(n+\frac{3N}{4}\right) \right] W_N^{n} W_{N/4}^{kn} \\
X(4k+2) &= \sum_{n=0}^{N/4-1} \left[x(n) - x\left(n+\frac{N}{4}\right) + x\left(n+\frac{N}{2}\right) - x\left(n+\frac{3N}{4}\right) \right] W_N^{2n} W_{N/4}^{kn} \\
X(4k+3) &= \sum_{n=0}^{N/4-1} \left[x(n) + jx\left(n+\frac{N}{4}\right) - x\left(n+\frac{N}{2}\right) - jx\left(n+\frac{3N}{4}\right) \right] W_N^{3n} W_{N/4}^{kn}
\end{aligned}
$$

- 위와 같이 N-pt DFT를 분해할 수 있습니다.
- twiddle factor를 상수로 바꿀수 있습니다.

Radix 4 Decimation in Frequency FFT(cont)



- 식을 그림처럼 바꿀 수 있습니다.
- 이는 4-pt, N/4-pt divde and conque에 해당합니다. 연산성능을 더욱 높이기 위해 해당 divde and conquer를 계속 적용하면 됩니다.
- Stage가 4096-pt의 경우 6으로 줄어들어 연상성능을 향상시킬 수 있습니다.

Radix 4 Decimation in Frequency FFT(cont)

```
for (iCnt1 = 0; iCnt1 < M_4; ++iCnt1)
{
    iQ = 0;
    for (iCnt2 = 0; iCnt2 < iM; ++iCnt2)
    {
        iA = iCnt2;
        fReal_Wq  = W_in[    iQ].real;
        fImag_Wq  = W_in[    iQ].img;
        fReal_W2q = W_in[2 * iQ].real;
        fImag_W2q = W_in[2 * iQ].img;
        fReal_W3q = W_in[3 * iQ].real;
        fImag_W3q = W_in[3 * iQ].img;

        for (iCnt3 = 0; iCnt3 < iL; ++iCnt3)
        {
            iB = iA +     iM;
            iC = iA + 2 * iM;
            iD = iA + 3 * iM;
```

- M_4는 stage 수 입니다.
- iM은 각 stage의 DFT단에서의 크기에 해당합니다.
- iQ는 그래프 상에서 곱해지는 각 twiddle_factor의 index 해당합니다.
  - 식과 그래프에 따라 n = 0 ~ N/4 까지 커지면서 곱해지는 twiddle factor도 바뀌는 것을 확인 할 수 있습니다.
- iA, iB, iC, iD는 각 DFT단에서 4-pt DFT를 위한 요소의 index에 해당합니다.

$$X(4k) = \sum_{n=0}^{N/4-1}\left[x(n) + x\left(n+\frac{N}{4}\right) + x\left(n+\frac{N}{2}\right) + x\left(n+\frac{3N}{4}\right)\right] W_N^{0} W_{N/4}^{kn}$$

$$X(4k+1) = \sum_{n=0}^{N/4-1}\left[x(n) - jx\left(n+\frac{N}{4}\right) - x\left(n+\frac{N}{2}\right) + jx\left(n+\frac{3N}{4}\right)\right] W_N^{n} W_{N/4}^{kn}$$

$$X(4k+2) = \sum_{n=0}^{N/4-1}\left[x(n) - x\left(n+\frac{N}{4}\right) + x\left(n+\frac{N}{2}\right) - x\left(n+\frac{3N}{4}\right)\right] W_N^{2n} W_{N/4}^{kn}$$

$$X(4k+3) = \sum_{n=0}^{N/4-1}\left[x(n) + jx\left(n+\frac{N}{4}\right) - x\left(n+\frac{N}{2}\right) - jx\left(n+\frac{3N}{4}\right)\right] W_N^{3n} W_{N/4}^{kn}$$

Radix 4 Decimation in Frequency FFT(cont)

```
fRealA = input[iA].real + input[iB].real
       + input[iC].real + input[iD].real;
fImagA = input[iA].img + input[iB].img
       + input[iC].img + input[iD].img;
fRealB = input[iA].real + input[iB].img
       - input[iC].real - input[iD].img;
fImagB = input[iA].img - input[iB].real
       - input[iC].img + input[iD].real;
fRealC = input[iA].real - input[iB].real
       + input[iC].real - input[iD].real;
fImagC = input[iA].img - input[iB].img
       + input[iC].img - input[iD].img;
fRealD = input[iA].real - input[iB].img
       - input[iC].real + input[iD].img;
fImagD = input[iA].img + input[iB].real
       - input[iC].img - input[iD].real;

input[iA].real = fRealA;
input[iA].img = fImagA;
input[iB].real = fRealB * fReal_Wq  - fImagB * fImag_Wq;
input[iB].img = fRealB * fImag_Wq  + fImagB * fReal_Wq;
input[iC].real = fRealC * fReal_W2q - fImagC * fImag_W2q;
input[iC].img = fRealC * fImag_W2q + fImagC * fReal_W2q;
input[iD].real = fRealD * fReal_W3q - fImagD * fImag_W3q;
input[iD].img = fRealD * fImag_W3q + fImagD * fReal_W3q;
```

- iA, iB, iC, iD index에 따라 실제 연산이 수행되는 부분입니다.
- 그래프의 butterfly는 축약된 형태로 실제로는 더해지기 이전에 곱해지는 상수가 다릅니다. 그래서 식을 참고해서 각각 4가지의 다른경우에 대한 연산을 진행해야합니다.
- 앞 슬라이드에서 계산된 twiddle factor와 곱한후 그대로 해당하는 자리에 넣어주면 됩니다.

$$X(4k) = \sum_{n=0}^{N/4-1}\left[x(n) + x\left(n+\frac{N}{4}\right) + x\left(n+\frac{N}{2}\right) + x\left(n+\frac{3N}{4}\right)\right] W_N^{0} W_{N/4}^{kn}$$

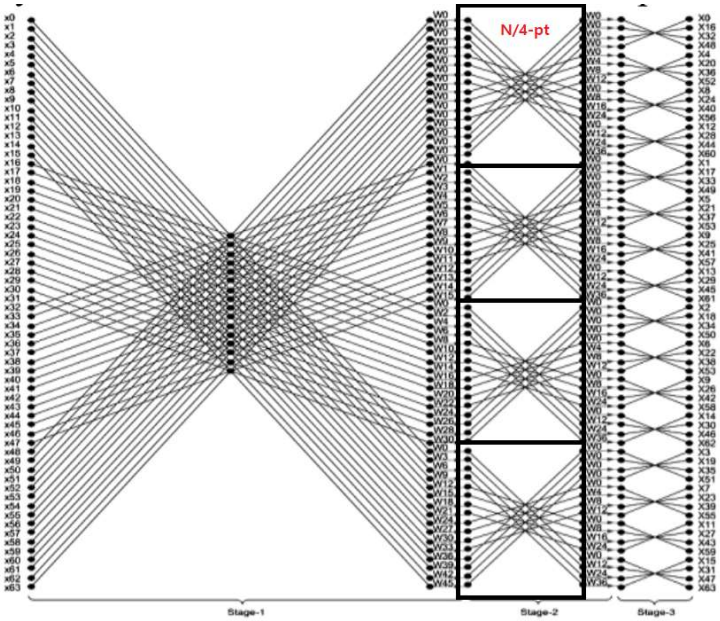
$$X(4k+1) = \sum_{n=0}^{N/4-1}\left[x(n) - jx\left(n+\frac{N}{4}\right) - x\left(n+\frac{N}{2}\right) + jx\left(n+\frac{3N}{4}\right)\right] W_N^{n} W_{N/4}^{kn}$$

$$X(4k+2) = \sum_{n=0}^{N/4-1}\left[x(n) - x\left(n+\frac{N}{4}\right) + x\left(n+\frac{N}{2}\right) - x\left(n+\frac{3N}{4}\right)\right] W_N^{2n} W_{N/4}^{kn}$$

$$X(4k+3) = \sum_{n=0}^{N/4-1}\left[x(n) + jx\left(n+\frac{N}{4}\right) - x\left(n+\frac{N}{2}\right) - jx\left(n+\frac{3N}{4}\right)\right] W_N^{3n} W_{N/4}^{kn}$$

Radix 4 Decimation in Frequency FFT(cont)

```
            iA = iA + 4 * iM;
        }
        iQ += iL;
    }
    iL *= 4;
    iM /= 4;
}
```



- 한 stage마다 N/4-pt DFT단이 x4씩 늘어납니다.
- 모든 DFT단에 대해서 연산을 진행하기위해서 iA를 그다음 DFT단으로 이동시켜서 연산을 진행합니다.
- iQ는 twiddle factor의 index로 stage가 증가하면서 더 크게 증가합니다.(N->N/4) 이를 반영하기 위해서 iL을 더해줍니다.
  - iL은 매 stage마다 x4씩 증가합니다.
- iM은 stage가 증가할때 DFT단의 크기는 1/4로 감소하므로 나눠줍니다.

$$X(4k) = \sum_{n=0}^{N/4-1}\left[x(n) + x\left(n+\frac{N}{4}\right) + x\left(n+\frac{N}{2}\right) + x\left(n+\frac{3N}{4}\right)\right] W_N^0 W_{N/4}^{kn}$$

$$X(4k+1) = \sum_{n=0}^{N/4-1}\left[x(n) - jx\left(n+\frac{N}{4}\right) - x\left(n+\frac{N}{2}\right) + jx\left(n+\frac{3N}{4}\right)\right] W_N^n W_{N/4}^{kn}$$

$$X(4k+2) = \sum_{n=0}^{N/4-1}\left[x(n) - x\left(n+\frac{N}{4}\right) + x\left(n+\frac{N}{2}\right) - x\left(n+\frac{3N}{4}\right)\right] W_N^{2n} W_{N/4}^{kn}$$

$$X(4k+3) = \sum_{n=0}^{N/4-1}\left[x(n) + jx\left(n+\frac{N}{4}\right) - x\left(n+\frac{N}{2}\right) - jx\left(n+\frac{3N}{4}\right)\right] W_N^{3n} W_{N/4}^{kn}$$